

بسم الله الرحمن الرحيم



تعلم لغة جافا للمبتدئين



الإصدار الأول

إعداد وتنفيذ : علي أبو سيفين



مقدمة عامة حول Java

ظهرت لغة جافا عام 1995 حيث طورتها شركةSun Microsystems ، وهي لغة سهلة الاستعمال قد صممت لتكون مستقلة عن محيط التشغيل وعبر عن ذلك شركة Sun بالمبدأ القائل

"Write once, Run Everywhere"أي" اكتب مرة واحد، وشغل أينما تريد"

يتطلب تحقيق هذا المبدأ تعريف وتحقيق ما نسميه بآلة جافا الافتراضية "Java Virtual Machine"أو اختصارا JVM إن آلة جافا الافتراضية تحاكي عمل معالج له تعليماته الخاصة وبحيث يسمح تحقيق هذه الآلة على منصات عمل متنوعة بنقل برنامج جافا من محيط تشغيل إلى آخر وتشغيلها دون الحاجة إلى تهديلها. أي أن أي برنامج محقق بلغة جافا يمكن أن يعمل على منصة Windowsيمكن تشغيله على منصة Linux مثلا) دون الحاجة لتعديل أي شيء في هذا البرنامج.

إن لغة جافا سهلة الاستعمال على مختلف منصات العمل (كما ذكرت سابقا) وتعتبر اللغة المثلى لتوزيع البرامج التنفيذية على الويب Web وتسمح جافا إضافة إلى إنشاء تطبيقات مستقلة بإنشاء برمجيات جافا (Java Applets) التي تملك نفس الوظائف الأساسية للتطبيقات إضافة إلى قدرتها على العمل ضمن متصفح ويب متوافق مع جافا.

طبعا مجال البرامج التطبيقية الموزعة تجاوز حاليا مجال الويب والانترنت (وسأترك الشرح في هذا المجال للشباب، حتى يقدموا لنا شرحا وافيا عن هذا المجال وأهمية لغة الجافا فيه).

من المميزات الخاصة في جافا أنها تتعامل مع العناصر باستخدام المؤشرات وهي لا تسمح بإنشاء مؤشرات خارج نطاق ترميزها الخاص، وفي جافا تقع مسؤولية تحرير مواقع الذاكرة التي تشغلها أغراض غير مستخدمة على عاتق مجمع النفايات (Garbage Collector) لذلك فإن برامج جافا لا تنقل الفيروسات. كما



وأن جافا لغة قابلة للتوسع بدون حدود فهي تعرف صفوف الأغراض باشتقاقها من صفوف أخرى موجودة ولها جميعها أب مشترك وحيد هو الصفObject

لقد وضعت شركة Sun في هذه اللغة خصائص مميزة جدا وزودتها بعدة مكونات أذكر منها:

1 – مجموعة أدوات التطوير (Java Development Kit) أو اختصارا JDK والتي تتضمن :

-javac.exe وهو المترجم الذي يقوم بتحويل البرنامج المصدر إلى الترميز الوسيط (byte code) الذي يستطيع المفسر تنفيذه.

-java.exe وهو مشغل التطبيقات أي البرنامج الذي يسمح بتنفيذ الترميز الوسيط المكون للتطبيق وفي الواقع يمكن تنفيذ الترميز الوسيط بعدة طرق إذ يمكن ترجمة هذا الترميز إلى لغة المعالج الحقيقية بواسطة المترجم (Just-in-time) أو JIT كما يمكن أيضا تفسيره بواسطة مفسر ويقتصر دور البرنامج java.exe على شحن الترميز إلى الذاكرة ومن ثم يتم عمل المترجم أو المفسر وذلك تبعا لكيفية تهيئة محيط العمل.

-applet Viewer الذي يسمح بتنفيذ البرمجيات(Applets)

-javadoc.exe الذي يولد توثيقا (Documentation) آليا لبرامج المستخدم بصيغة ملفاتHTML .

2 - محيط تشغيل جافا (Java Runtime Environment) أو JER الذي يتضمن كل ما يلزم لتشغيل ولتوزيع تطبيقات جافا على المستخدمين .



مميزات لغة جافا

1- لغة تلتزم بقواعد البرمجة بواسطة الأهداف Object Oriented Programming (OOP) : حيث وفرت كثير من الجهد الذي كان يبذل باستخدام البرمجة التقليدية ، حيث كانت البرمجة التقليدية توفر للمبرمج مكتبة من الدوالي إضافة إلى تركيب تقليدي للبرنامج وعلى المبرمج أن يستعمل الدوالي مع تركيب البرنامج لإنشاء التطبيقات مما يضطره لكتابة السطور الكثيرة أكثر من مرة ؛ لقد كانت وحدة بناء البرنامج هي الدالة .. في حين أتت البرمجة بواسطة الأهداف بفكرة جديدة هي إنشاء عناصر متكاملة تحتوي على بيانات ودوالي هي أساس إنشاء البرنامج .. وبالتالي أصبحت وحدة بناء البرنامج وحدة كبيرة هي الفصيلة أو العنصر Object مما سهل واختصر الكثير.

2- لغة لها بيئة تنفيذ خاصة : JVM للغة الجافا بيئة تشغيل للبرنامج هي JVM التي تقوم بترجمة البرنامج للغة الآلة وبالتالي فإن لغة الجافا غير مرتبطة بنظام التشغيل.

3- لها مكتبة فصائل قوية: Class Libraries نظراً لأن لغة جافا تعتمد على مفهوم OOPفهي تحتوي على مكتبة فصائل قوية توفر معظم أو كل الفصائل المطلوبة للإعمال مثل التعامل مع الملفات وقواعد البيانات والشبكات و الرسومات المجسمة والحركة وكذلك التعامل مع الإنترنت.

4- لغة مبنية على لغة الـــ : C,C++فعندما تم إنشاء لغة الجافا كان أساس بنائها لغة من أشهر وأقوى اللغات وهي C,C++ وبالتالي فهي لم تبدأ من حيث بدأ الآخرون بل من حيث انتهى الآخرون وهي لغة C++ و ثم إضافة الجديد في لغة الجافا.



شرح لبعض مصطلحات لغة جافا

1- بيئة التشغيل : "JVM"

الحروف JVM اختصار للعبارة , JAVA Virtual Machine وهي فكرة قامت جافا بإنشائها لتجعل لغة جافا تعمل على جميع أو معظم أنظمة التشغيل . وتقوم الفكرة على إنشاء طبقة وسيطة Software كأنها برنامج تشغيل للبرامج Runtimeلكل نظام تشغيل يتم إنزاله أولاً على الأجهزة بحيث تفهم هي برامج جافا وتفسرها لنظام التشغيل ثم الجهاز ولهذا كان من مزايا لغة جافا أنها تعمل على كثير من نظم التشغيل الموجودة بعد إعداد JVM الخاصة بمعظم أنظمة التشغيل .. فلا يهم إذا كان البرنامج مكتوب لنظام التشغيل WINDOWS أو UNIX , المهم أن البرنامج يكتب ثم يحمل إلى الجهاز وعلى الجهاز يوجد JVM للنظام الموجود وبالتالي يعمل البرنامج .

Java Applet -2

نوع من أنواع التطبيقات الذي صمم خصيصاً للإنترنت حيث يقوم المطوّر بإعداد هذا البرنامج Applet ثم يستدعيه من خلال استخدام ملف HTML بشرط تحميل برنامج Applet على الخادم server الموجود عليه ملف الـــ HTML . أما طريقة إنشاء Applet وطريقة استدعائها من داخل ملف HTML فهذا ما سنتعلمه إن شاء الله خلال الدروس القادمة .

Java Application -3تطبيق الجافا:

هو تطبيق يشبه التطبيقات المنشأة بجميع لغات البرمجة الأخرى يعمل مع نظام التشغيل بعيداً عن شبكة الإنترنت والمشهور عن لغة جافا أنها تعدّ برامج للإنترنت ولكن غير المشهور أيضاً أنها توفر كثير من نقاط القوة في إعداد أي تطبيق سواء مكتبي DISKTOP أو خاص بالشبكات CLIENTSERVER.



القواعد الرئيسية للغة جافا

التعابير

التعابير هي أساس أي شفرة برمجية ، بالتعاون مع الأساسيات الأخرى للغة جافا نستخدم التعابير لحساب قيم المتغيرات وتحليل النتيجة وذلك حتى نستطيع التحكم في طريقة سير وعمل البرنامج. ويتم ذلك عن طريق حساب القيمة وإرجاعها للكمبيوتر للقيام بفعل معين.

باختصار التعابير هي عبارة عن مجموعة متغيرات ومشغلات وأوامر لحساب قيمة معينة.

إذا عندما نقول ما هو التعبير لهذا الشي فإننا نقصد ما هي القيمة لحساب هذه العملية أو ما هي القيمة لهذا المشغل ..الخ.

مثلا التعبير لإضافة رقم واحد للمتغير يكون كالتالي :

```
count++;
```

فيقوم الكمبيوتر في هذه الحالة بحساب قيمة المتغير countثم يضيف إليها الرقم واحد.

والتعبير لضرب ثلاث متغيرات ببعضها البعض يكون كالتالي :

x * y * z

وانتبه للترتيب ، تحسب أولاً قيمة المتغير x ثم تضرب بقيمة المتغير yبعد حسابها ثم تضرب النتيجة بقيمة المتغير z بعد حسابها وأخير يرجع التعبير القيمة الأخيرة للكمبيوتر.



ويلخص الجدول التالي مجموعة التعابير في لغة جافا :

<table>
<tr><td>postfix operators</td><td>[] . (params) expr++ expr--</td></tr>
<tr><td>unary operators</td><td>++expr --expr +expr -expr ~ !</td></tr>
<tr><td>creation or cast</td><td>new (type)expr</td></tr>
<tr><td>multiplicative</td><td>* / %</td></tr>
<tr><td>additive</td><td>+ -</td></tr>
<tr><td>shift</td><td><< >> >>></td></tr>
<tr><td>relational</td><td>< > <= >= instanceof</td></tr>
<tr><td>equality</td><td>== !=</td></tr>
<tr><td>bitwise AND</td><td>&</td></tr>
<tr><td>bitwise exclusive OR</td><td>^</td></tr>
<tr><td>bitwise inclusive OR</td><td>|</td></tr>
<tr><td>logical AND</td><td>&&</td></tr>
<tr><td>logical OR</td><td>||</td></tr>
<tr><td>conditional</td><td>? :</td></tr>
<tr><td>assignment</td><td>= += -= *= /= %= &= ^= |= <<= >>= >>>=</td></tr>
</table>



المتغيرات

تستخدمها للتعبير عن معلومات متغيرة خلال عمل البرنامج ، ولكل متغير نوع و اسم و مجال كما في الجدول التالي :

النوع	Format/الحجم	المواصفات
الأرقام الصحيحة		
byte	8-bit two's complement 8 بت من مضاعفات العدد 2	
Byte-length integer		
short	16-bit two's complement	Short integer نستخدمه للأعداد الصحيحة الأقل أو المساوية للعدد 2 مرفوع للأس 16
int	32-bit two's complement	Integer للأعداد الصحيحة الأقل أو المساوية للعدد 2 مرفوع للأس 32 بت
long	64-bit two's complement	Long integer للأعداد الصحيحة الأقل أو المساوية للعدد 2 مرفوع للأس 64 بت
الأرقام العشرية – الحقيقة		
float	32-bit IEEE 754	Single-precision floating point الأرقام العشرية الأقل أو المساوية للعدد 2 مرفوع للأس 32 بت
double	64-bit IEEE 754	Double-precision floating point الأرقام العشرية الأقل أو المساوية للعدد 2 مرفوع للأس 64 بت



أنواع أخرى		
char	16-bit Unicode character	A single character حرف واحد
boolean	true or false	A boolean value (`true` or `false`) قيمة "بولن" صحيح أو غير صحيح

مثال :-

```
Import java.io.*;
Public class Count {
Public static void countChars(Reader in)
throws IOException
            {
      int count = 0;

      while (in.read()  != -1)
      count++;
     System.out.println(“Counted “ + count + “
chars.”);
            }
                              //..main method omitted..
```



المصفوفات والسلاسل

في كثير من الأحيان ، نحتاج إلى طريقة للتعامل مع مجموعة أرقام في وقت واحد في برنامجنا والنوع المناسب لذلك هو ما يعرف بـــ Array ، نحن في قسم المتغيرات تعرفنا على مجموعة أنواع ولكننا فصلنا النوعArray لان له استعمالات خاصة. وكما هو الحال مع الأرقام ، أحيانا كثيرة نود أن نتعامل مع مجموعة حروف (قد تمثل هذه الحروف جملة معينة أو اسم معين) ولهذا نستخدم النوع.Strings

Arrays

وكما كان الحال مع بقية أنواع المتغيرات فأننا يجب أولا أن نقوم بتعريفها في برنامجنا حتى نستطيع الاستفادة منها. وبما أن Arrayهو عبارة عن مجموعة أرقام فأن هذه الأرقام قد تكون من مختلف أنواع الأرقام التي اطلعنا عليها سابقا (صحيحة ، حقيقية ، عشرية ...الخ).

لننظر للمثال التالي :

```
int[] arrayOfInts;

int[] arrayOfInts = new int[10]

for (int j = 0; j < arrayOfInts.length; j ++)

{

arrayOfInts[j] = j;

System.out.println("[j] = " + arrayOfInts[j]);

}
```



قمنا في السطر الأول بتعريفه (جعلناه من نوع الأرقام الصحيحة).

بعد ذلك قمنا بتحديد حجمه (في هذه الحالة 10)

ثم استخدمنا حجمه في عملية التكرار (عرفنا الحجم عن طريق(length).

ثم أضفنا لكل عنصر فيه قيمة معينة

وأخيرا قمنا بطباعة تلك القيمة على الشاشة.

ملاحظة : النوع Arrayعبارة عن مؤشر لكائن ، وبما انه مؤشر هناك مجموعة أعضاء نستطيع استخدامها ، مثل lengthلإيجاد حجمه كما فعلنا في المثال السابق. وكذلك استخدمنا الأمر new لحجز جزء من الذاكرة له.

Strings

هذا النوع شبيه بالنوع السابق ولكنه يتعامل مع الأحرف بدل الأرقام.

لننظر للمثال التالي :

```
String[] arrayOfStrings = new String[10];
for (int i = 0; i < arrayOfStrings.length; i ++)
 {
arrayOfStrings[i] = new String("Hello " + i);
}
```



قمنا في السطر الأول بتعريفه وتحديد حجمه (في هذه الحالة 10)

ثم استخدمنا حجمه في عملية التكرار (عرفنا الحجم عن طريق .length)

ثم أضفنا لكل عنصر فيه قيمة معينة.

طبعا هذا الشرح البسيط لا يسمن ولا يغني من جوع لأن لغة جاف لغة دسمة ولكني أحببت فقط أن أعطيك نوع من الانطباع الأولي عن هذه النوعية من المتغيرات ولترى بنفسك الأوامر المتوفرة لهذا النوع .

أوامر التحكم في سير البرنامج

لننظر أولا للجدول التالي :

المعنى	الأوامر
صنع القرار	if-else, switch-case
التكرار	for, while, do-while
الأخطاء	try-catch-finally, throw
أوامر مختلفة	break, continue, label: , return



الآن لابد انك تتساءل ماذا تعنيه كل هذه الكلمات :

صنع القرار: تعني الأوامر التي نستطيع من خلالها اتخاذ قرار معين في البرنامج ، مثلا نقول إذا كان المتغير Xيساوي 5 نضع المتغير Yليساوي 10و هكذا.

التكرار : تعني الأوامر التي من خلالها ندخل في دورة معينة ، مثلا كلما كان المتغير X اصغر من 10 يزيد معدل قيمة المتغير Y بمعدل 1 وقيمة Xبمعدل 1 أيضا.

الأخطاء: وتعني الأوامر التي عن طريقها نستطيع أن نعرف الخطاء في البرنامج إذا وقع.

طيب ، بعدما شرحنا ما تعنيه هذه الكلمات أنا متأكد انك لا زلت في حيرة منها أو بالأحرى في حيرة من الأوامر التي تمثلها وهذه الحيرة ستختفي بطبيعة الحال حالما ترى مثال لكل أمر من تلك الأوامر. وسنحاول هنا شرحها ببعض التفصيل.



أوامر صنع القرارات

الأمر إذا كان – أو if - else

```
if ( تعبير )

   Statement

else

   Statement
```

وتعني إذا كان (التعبير) صحيحاً فقم بالأوامر التالية Statementأو أن لم يكن التعبير صحيحا فقم بالأوامر التالية Statement. وكلمة Statement تعني أي شفرة برمجية أنت تكتبها.



مثال (افرض انك تريك من الكمبيوتر أن يعطي الطلبة في صف معين مستواهم حسب العلامات الحاصلين عليها في الامتحان ، لنستخدم المتغير testscore ليعبر عن علامة كل طالب ثم لنستخدم المتغير gradeليعبر عن المستوى) :

```
int testscore;
char grade;

if (testscore >= 90) {
grade = 'A';
} else if (testscore >= 80) {
grade = 'B';
} else if (testscore >= 70) {
grade = 'C';
} else if (testscore >= 60) {
grade = 'D';
} else {
grade = 'F';
}
```



الأمر المتغير – في حالة switch - case

نستخدم هذا الأمر لاختبار قيمة متغير معين وعلى أساسها نقوم في التحكم في سير برنامجنا.

لنأخذ مثال على ذلك : افرض أن في برنامجنا يوجد متغير سميناه monthيحتوي على القيمة 1 الى 12 والتي تعبر عن أشهر السنة ، ثم أننا نود من الكمبيوتر أن يكتب لنا الشهر الصحيح بناء على قيمة المتغير monthفإذا كان مثلا يساوي 4 نعرف أن الكمبيوتر المفروض أن يكتب لنا الشهر Aprilأو أبريل وهو الشهر الرابع في السنة. ونرى كيف أن هذا الأمر هو الأكثر مناسبة كما يلي :

```
int month;
switch (month) {
case 1:  System.out.println(“January”); break;
case 2:  System.out.println(“February”); break;
case 3:  System.out.println(“March”); break;
case 4:  System.out.println(“April”); break;
case 5:  System.out.println(“May”); break;
case 6:  System.out.println(“June”); break;
case 7:  System.out.println(“July”); break;
case 8:  System.out.println(“August”); break;
case 9:  System.out.println(“September”);
break;
case 10: System.out.println(“October”); break;
case 11: System.out.println(“November”);
break;
case 12: System.out.println(“December”);
break;
}
```



أوامر التكرار

الأمر من وإلى for

for (البداية; النهاية; التغيير معدل)

    statements

حيث يقوم الكمبيوتر بزيادة القيمة من البداية للنهاية على حسب معدل التغيير

، لنأخذ المثال التالي:

```
int i;
for (i = 0; i < 50; i++) {
    . . .
    // معين هنا قم بشيء
    . . .
}
```

في هذا المثال نرى أن الكمبيوتر سوف يقوم بتكرار مابين القوسين {} إلى أن تصبح قيمة المتغير i تساوي 50 .



الأمر مادام *while*

```
while (تعبير)
        statement
```

مادام تعبير معين صحيح إذا نفذ statement التالية.

لننظر للمثال التالي:

```
while (X != 10) {
   X++;
   System.out.println("X does not equal to 10 yet ");
}
```

الأمر افعل – مادام do –while

```
do {
   statements
} while (booleanExpression);
```

وتعني افعل statementsمادام التعبير booleanExpression صحيح.



لنرى المثال التالي :

```
int c;
. . .
do {
   c-- ;
   . . .
} while (c != -1);
```

أوامر مختلفة

الأمر   توقف   break

لو نظرت إلى الأمر switch - case في الأعلى فأنك ستلاحظ استخدامنا لأمر التوقف breakفي المثال التابع له وكان عملها الذهاب للحالة التالية.

هناك وظيفة أخرى لأمر التوقف وهو الذهاب لجزء من الشفرة البرمجية علمناها



باسم معين ثم نتبع ذلك الاسم بـــنقطتين : ثم بأمر معين كما يلي :

```
breakToHere: someJavaStatement
```

كما نرى اخترنا علامة سميناها breakToHere متبوعة بالنقطتين : ثم بأي أمر نشاء بعد ذلك في موقع مختلف نستطيع القفز لمكان العلامة كما يلي:

```
break breakToHere;
```

ملاحظة : هذا الأمر شبيه بالأمر goto في لغة والسي المحسنة و الغير مستخدم في جافا.



الوراثة Inheritance

قد نحتاج في بعض الأحيان إلى الوراثة في البرمجة وخاصة في مصطلحات oop.

فنحن عندما نقوم بوراثة صنف "class" نقوم بوراثة صفاته العامة دققوا على كلمة عامة public ....

بشكل عام أي صنف " class " تستطيع أن تورثه بشرط أن يكون عام مجال الرؤية له اي public

نسمي الصنف الذي يورث اي الصنف الذي يعطي مواصفاته بالصنف الأب super class

اما الصنف الذي يرث من صنف أخر نسميه subclass

في البرامج الكبيرة يشترك أكثر من مبرمج في كتابة البرنامج ومن الممكن أن يكتب أحد المبرمجين abstract classes أي class تجريدية لا يحوي جسمها أي تحقيق لشئ فقط بروتوكول عام ..فيقوم المبرمجين الآخرين بصناعة subclass ليحققوا abstract classes في method معينة abstract classes (تجدونه في احد مقالات المدونة)

يجب أن نعرف أيضا موضوع طريقة الاشتقاق أو الوراثة حيث نستخدم الكلمة المفتاحية extends قبل أن نضع اسم subclass

```
public class SUPER {
public int x;
public int y;
protected String name;
public void method()
{
…….
}
}//end super class
class SUBCLASS extends SUPER
{
public static void main( String [] arg) {
//can use any parmter or method by class supper
{
{//end subclass
```



من مفاهيم الـــ opp: الوراثة – تعدد الأشكال .

المفتاح الرئيسي للبرمجة الغرضية التوجه هو الوراثة:أي إعادة خلق صفوف جديدة(ابن) من صفوف قديمة(أباء) ,لها نفس الخصائص الموجودة في الصف القديم وخصائص جديدة أخرى خاصة بالصف الابن.

يطلق على الصف الأب اسمsuperclass

يطلق على الصف الابن اسم subclass

تتم الوراثة عن طريق الكلمة extends.

الوراثة في الجافا وراثة أحادية وليس هناك وراثة متعددة الأشكال (يعني لكل صف ابن أب مباشر واحد).

ملاحظات :

أولاً:

sub object is a super object

يعني أن أي غرض من الصف الابن هو غرض من الصف الأب والعكس غير صحيح أي:

super object is not a sub object

ثانياُ:الصف الابن يمكنه الوصول إلى أعضاء الأب إذا كان من النوع public

الصف الابن لا يمكنه الوصول لأعضاء الأب إذا كانت من النوع private.

أما إذا كانت أعضاء الأب من النوع package access يستطيع الابن الوصول إليه إذا كان الابن عضو في نفس الـــ package .

أما إذا كانت protected :يمكن الوصول إليها من قبل الصف الأب و الصف الابن والصفوف الموجودة في حزمة الصف الأب.



ثالثاُ:

مفهوم الـــ over riding :أي إعادة صياغة الطريقة الموجودة في الصف الأب داخل الصف الابن والسبب في ذلك أنه قد تكون الصيغة في الطريقة الأب غير مناسبة للصف الابن.

رابعاُ:

المعامل super يشير إلى الصف الأب للصف الحالي .

خامساُ:

مفهوم الـــ is a : تعني الوراثة الكاملة .

مفهوم الـــ has a :تمثل استدعاء الصف ومن ثم استخدام غرض من الصف المستدعى ضمن الصف الذي نكتبه .

سادساُ:

يمكن تحويل الغرض من الصف الأب إلى الغرض من الصف الابن وذلك عن طريق الـــ casting القسري , وإذا لم نضع الـــ casting سينتج لدينا استثناء أي :exeption .

سابعاُ:

الصف الابن لا يرث من الصف الأب الباني حتى نضع في باني الابن المعامل super. أي نفذ باني الأب ,إذا لم نضع التعليمة super داخل الباني الابن سيتم استدعاء الباني الافتراضي للأب(الذي يهيئ المتحولات بقيم صفرية للأولية ,false للبوليانية ,null للمرجعية) وإذا لم يكن هناك باني افتراضي للأب سيقوم البن باستدعاء الباني الافتراضي للصف object (الأب الروحي للجافا).



*- الطرق والصفوف من النوع final :

1- الصف من نوع final لا يمكن وراثتها أبداُ (لا يمكن أن يكون أب لأحد).

2- الصف من النوع الغير final : إذا كان فيه متحول من النوع final لا يمكن عمل الـــ over riding عليه في الصف الابن .2- الطرق من النوع static :هي فعلياُ تعتبر final .

الطرق من النوع private : هي أصلاُ لا تنزل مع الصف الابن وبالتالي هي final .

3- إذا كان الصف من النوع final فإن جميع الطرق فيه من النوع final .

5- تسريع الترجمة: عملية الـــ call تأخذ زمن أطول في الترجمة من كتابة الكود بذاته . فالمترجم عندما يرى استدعاء للطريقة من النوع final يضع بدلاُ عنه الكود بذاته وذلك لأنه لا يتغير .

*- finalizer:

كيف نتعامل مع sub و super عند استدعاء الـــ finalizer (هو طريقة موجودة في الصف object فقبل تحرير الذاكرة من قبل ال gc فإنه يتم استدعاء الطريقة finalizer)

إذا لم نضع أي inalizer وكان لدينا وراثة متعددة فليس هناك مشكلة.

وإذا وضعنا ال finalizer في أحد الصفوف فإن ابنه يجب أن يأخذ الـــ finalizer من أبوه ونضع في الابن كما يلي :

(super.finalizer)



الاستثناءات Exceptions

الاستثناءات في جافا Exceptions هو غرض يتم توليده عند حدوث وضع غير طبيعي في برنامجك
هذا الغرض يمتلك حقولا Data Members تقوم بتخزين معطيات تعبر عن طبيعة المشكلة الناتجة عن
)الوضع غير الطبيعي.(

عملية قذف (رمي) الاستثناء Throwing Exception

إن الغرض المعرف للظرف الاستثنائي كوسيط argument يجب أن يتم رميه (قذفه (إلى جزء من كود برنامجك تم كتابته خصيصا للتعامل مع هذا النوع من المشاكل.
إن الكود الذي يستلم غرض عن الصف Exception أو احد أبناءه كوسيط (بارامتر) نقول انه التقطه
catch it

بفرض أن س= الاستثناءات (الأغراض) الناتجة عن صفوف ترث الصف Exception أحد أبنائه) عدا الصف RuntimeException

إذا كان هناك كود في برنامجك نظن انه سينتج عنه س.
إذا كان هناك طريقة ما في برنامجك method تظن أنها تقوم بتوليد احد أنواع س.
فإنه سيكون لديك خيارين (يمكنك تطبيق الخياران معا:)
الخيار الأول : التقاط الاستثناء ضمن ال method أي catch it
تحويط الكود المشكوك فيه ضمن كود ال method بكتلة try
وحل المشكلة (في حال حصلت) ضمن كتلة catch



كود

try{الكود المشكوك بأنه يقذف استثناء}
catch(الغرض المناسب لطبيعة الاستثناء) { الكود الذي يجري تنفيذه إذا حصل وحدث الاستثناء }

الخيار الثاني : تتهرب من التقاط الاستثناء (لا نضع كتلتي try-catch)
و تقدم إشعارا بأن هذه الطريقة قد ترمي (تقذف) استثناء ما
ويرفع مستوى معالجة الاستثناء إلى مستوى أعلى (مستوى الصف الذي يستدعي هذه الطريقة)
i
كود

```
int myMethod() thorows EOFException {…….}
```

الصف الذي يستدعي الطريقة: myMethod()

كود

```
public MyClass{
…..
try{ int x= myMethod() ; }
catch(Excrption e){ ….. }
…..
}
```



الاستثناء RuntimeException

هناك مجموعة من الاستثناءات (أغراض ناتجة عن صفوف ترث Exception) هي الصفوف التي ترث الصف RuntimeException الذي هو ابن للصف Exception.
هذه الاستثناءات يسمح لك المترجم Compiler بتجاهلها وعدم اختيار الاختياران السابقان بحيث تتم عملية الترجمة.
هذه الاستثناءات تظهر (بمعنى يتم توليد غرض من صف يرث RuntimeException) بسبب أخطاء معينة في الكود البرمجي بحيث يكون نوع الصف الذي نتج عنه نوع الغرض الاستثناء ملائم لنوع الخطأ أو المشكلة.
ومع ذلك يمكنك التقاطها Catch it وكتابة كود للتعامل معها وتصحيحها (استخدام الخياران السابقان(إن أردت ذلك) .

أمثلة لهذه الاستثناءات

صف الاستثناء : نوع المشكلة أو الحالة الاستثنائية التي يمثلها

ArithmeticException : الحالات الحسابية الغير مسموحة مثل محاولة القسمة على صفر .

ClassCastException : محاولة إجراء تحويل قسري على غرض من صف ما a مثلا ...إلى صف جديد ليس نفس الصف a ولا احد أبنائه ولا احد اباته.

ArrayStoreException : محاولة تخزين غرض من نمط ما في مصفوفة بحيث نمطه لا يناسب النمط المعرفة به المصفوفة.

NullPointerException : استعمال متحول غرض قيمته (null = لم تجرى له عملية new)لتمريره كوسيط ل method ما أو لباني ما أو إسناده ل data member.

IllegalArgumentException : تمرير برامتر إلى method لا يتلاءم نمطه مع نمط البارامتر الممرر.



لنعود إلى الاختيار الأول : التعامل مع الاستثناء في مكان حدوثه (التقاطه ومعالجته مباشرة حال حدوثه)

كود

}try
هنا كود قد يعطي استثناء أو أكثر (ممكن من انواع صفوف مختلفة)
إذا لم يعطي هذا الكود أي استثناء وتنفذ كله بنجاح نتجاهل كتلة Catch وننتقل لتنفيذ ما في كتلة finally
;Statment1
;Statment2
;Statment3
;Statment4
{

كود
catch(IOException e){
كود يعالج الاستثناء الذي جرى الكشف عنه في كتلةtry
يعالج الاستثناء من النوع المحدد بين القوسين حصرا (هنا (IOException أو أحد آبائه
حيث يجرى هنا اتخاذ الإجراءات اللازمة في حالة حدوث المشكلة ومن ثم الانتقال لكتلة finally
إذا لم يكن الاستثناء المقذوف في كتلة try من النوع IOException سيتم الانتقال لكتلة catchالتالية.
فإن لم يكن هناك catch تالية .. أما إن يكون الصف أو ال methodتقذف استثناء بالكلمة throws
)فائدة استخدام الخياران معا(
وان لم تكن تستخدم الخيار الثاني ... إما إن يكون الاستثناء من نوع ابن ل RuntimeExceptionويتجاهله المترجم.



وإما لا يكون ابن له وتحصل على أخطاء في زمن الترجمة ولن تتم ترجمة برنامجك.
}

كود

```
catch(Exception e){}
```

قد يكون الاستثناء الذي جرى قذفه ليس من النوع الذي حددته بين قوسين catch وقد يكون هناك أكثر من استثناء من أكثر من نوع والعمل ؟؟!!

ممكن أن نضع أكثر من كتلة catch متتالية بشرط ترتيبها الأبناء أولا ثم الإباء ... الصغير فالأكبر فالأكبر

(حسب شجرة الوراثة وإلا ستحصل على أخطاء في زمن الترجمة ولن تتم ترجمة برنامجك.)

انتهاء بكتلة catch (Exception e){} حيث Exception الصف الأب الأكبر لكل الاستثناءات.

التي ستقبل أي نوع من الاستثناءات في حال لم تستطيع أن تحزر كل الأنواع المناسبة للاستثناءات التي قد تحدث.

كود

```
finally{الكود الذي يلي تلك الأحداث كلها}
```

ملاحظة هامة:

بفرض أن تنفيذ ;Statment2 سينتج عنه قذف استثناء.

فإن ما يحصل هو تنفيذ ;Statment1 بنجاح ثم الانتقال إلى فحص كتل catch بحثا عن الاستثناء الملائم ومن ثم الانتقال إلى كتلة finally وهذا يعني أن كلا من

Statment3;
Statment4;

لن يجري تنفيذهما ويجب الانتباه الى هذه النقطة كي لا يتم فقد أجزاء مهمة من البرنامج.



ملاحظة : كتلة finally اختيارية وممكن عدم وضعها.

ملاحظة : إذا لم يقذف الكود ضمن try أي استثناء يجري تنفيذ كامل الكود ضمن try وتجاهل كل كتل catch والانتقال لتنفيذ كتلة finally أن وجدت.

متى نختار الاختيار الثاني:

قد نتهرب من معالجة الاستثناء مباشرة في مكانه لعدم معرفتنا بنوعه.

قد نشك إننا لم نضع الأنماط المناسبة في كتل (catch لم نستعمل catch(Exception (لعدم مراعاة الدقة) مما يؤدي إلى انه لم تتم معالجة الاستثناء (لم تلتقطه أي من كتل catch) مما يؤدي إلى أخطاء في زمن الترجمة.

أحيانا تظن أن أقصى ما تستطيع فعله لمعالجة الاستثناء بعد التقاطه في كتلة catch هو كتابة رسالة خطأ .

لكن قد تكون رسالتك معبرة أكثر ومفيدة أكثر إذا حوت معلومات عن طبيعة المشكلة الناتجة عن الاستثناء وسياقها حيث انه كما علمنا سابقا أن الغرض من الصف Exception (أو أحد أبناءه) الذي يتم تمريره إلى كتلة catch يحوي معلومات عن طبيعة المشكلة التي تسبب الاستثناء ....

للاستفادة من هذه المعلومات المخزنة في هذا الغرض علينا فهم بعض خصائص الصف Throwable الذي يرثه كل استثناء (الأب المباشر للصف Exception)

الباني الأساسي للصف Throwable له وسيط من نمط String

في حالة حدوث استثناء يمرر المترجم في زمن الترجمة إلى هذا الوسيط عبارة تعبر عن طبيعة المشكلة المسببة للاستثناء .

الأغراض من نمط الصف Throwable تحوي شيئان حاويان لمعلومات حول الاستثناء :

الرسالة المعبرة عن طبيعة المشكلة التي يتم تحميلها في الباني (تحدثت عنها سابقا)



سجل مكدس التنفيذ Record of execution Stack يتكون حال حدوث الاستثناء ....

هذا السجل يقتفي اثر الmethod المسببة لتوليد الاستثناء

(مكدس Stack : يعمل حسب مبدأ LIFO أخر من دخل هو أول من يخرج)

هذا المكدس يرجع بالوراء ليجد الكتلة التي استدعتها (قد يكون استدعاء بداخل استدعاء)

وبالتالي فإن سجل مكدس التنفيذ سوف يحتوي رقم سطر الكود يولد الاستثناء متبوعا باقتفاء اثر استدعاءات هذه الmethod حتى نحصل على الاسم الكامل لكل method تستدعي الmethod الحاوية للاستثناء متبوعا برقم سطر كود الاستدعاء .

كود

```
public class class3}

public static void main(String[] args) { method1{;()

{
```

كود

```
class class1}

method1 () { method2{;()

{
```



كود

class class2}

method2… } ()

كود حاوي يسبب استثناء

{

في المثال طبعا يبدأ الترجمة بقراءة main من class3 إلى أن يصل للاستدعاء method1()

....هذا أول استدعاء يحشره مكدس سجل التنفيذ

يترك المترجم class3 ذاهبا للبحث عن كود method1() في class1 فيجد ضمنه استدعاء method2()

يحشر سجل التنفيذ الاستدعاء هذا فوق الأول .

يترك المترجم class1 ذاهبا لترجمة كود method2() الموجود في class2 فيجد ان السطر 15 مثلا يولد استثناء .

يحشر المكدس رقم سطر الكود المسبب للاستثناء

ومن ثم يرسل بمحتوياته إلى الغرض Exception (أو احد أبناءه) حسب LIFO من الأخر للأول :

كود

PackageName.class2.method2: رقم سطر الكود الذي يولد الاستثناء



method2: PackageName.class1.method1 رقم سطر الكود الذي فيه استدعاء

method1 : PackageName.class3.main رقم سطر الكود الذي فيه استدعاء

أهم الـmethosd الموجودة في الصف Throwable :

getMessage() : تعيد محتوى الرسالة التي تصف الاستثناء غالبا يكون اسم الكلاس ووصف مختصر للاستثناء .

printStackTrace() : كتابة الرسالة ومحتويات مكدس التنفيذ على الخرج القياسي (شاشة الـconsole)

printStackTrace(PrintStream s) : نفس السابقة ولكن نحدد في البرامتر مجرى الخرج الذي سنكتب فيه الرسالة ومحتويات سجل مكدس التنفيذ .

fillInStackTrace() : تحديث مكدس اقتفاء الأثر إلى نقطة استدعاء هذه الـmethod حيث أن رقم السطر الحاوي للاستثناء سيصبح رقم سطر استدعاء هذه الـmethod وينسى الاستدعاءات السابقة .



الواجهات Interfaces

الواجهة هي نوع مرجعي تستخدمه الأنواع الأخرى لضمان أنها تدعم عملية معينة وهي تحدد عناصر معينة يجب أن تتضمنها الفئات التي تحقق هذه الواجهات وهي تحتوي على طرائق وخصائص وعناصر أحداث تماما كالفئات

الصيغة العامة

كود

```
[ <attributelist> ] [ accessmodifier ] [ Shadows ] _
Interface name [ ( Of typelist ) ]
   [ Inherits interfacenames ]
   [ [ modifiers ] Property membername ]
   [ [ modifiers ] Function membername ]
   [ [ modifiers ] Sub membername ]
   [ [ modifiers ] Event membername ]
   [ [ modifiers ] Interface membername ]فئة
   [ [ modifiers ] Class membername ]
   [ [ modifiers ] Structure membername ]
End Interface
```

وكما نرى من الصيغة العامة فيسبق التصريح عن الواجهة واصفات Attributes وكلمات تحديد المجال مثل Public أو الكلمة Shadows التي تعني أن هذه الواجهة تعيد تعريف واجهة موجودة وبنفس الاسم كما يمكن وراثة واجهة من أخرى تماما كالفئات وهي تحتوي على نفس العناصر الممكن احتوائها ضمن الفئات من وظائف ودالات وخصائص ... الخ ولكنها تحدد تعريف هذه الوظائف والخصائص فقط بدون الكود الذي يحدد عملها ويجب على أي فئة تحقق واجهة معينة أن توفر الكود العملي لكافة العناصر الموجودة ضمن هذه الواجهة ويمكن تعريف الواجهة على مستوى Namespace أو Module أي يجب أن يكون تعريفها عاما وغير محصور ضمن إجراء معين كما يمكننا تعشيش الواجهات بحيث أن أية واجهة ممكن أن تتضمن واجهة أخرى كما يمكن تحديد خاصية افتراضية باستخدام الكلمة Default ولا يمكن استخدام محددات الوصول مثل Public أو Private عند التصريح عن عناصر الواجهة ولكن يمكن استخدام Overloads أو Shadows وعندما تستخدم واجهة ضمن فئة

يتم الإعلان عن عناصر هذه الفئة باستخدام محدد الوصول Public افتراضيا الأمر الذي يمكنك تغييره لاحقا على مستوى تلك الفئة كما لا يمكن التصريح عن متغيرات ضمن الواجهة وعند تسمية الواجهات يفضل أن يبدأ الاسم دوما بالحرف I

لنرى الآن بعض الأمثلة عن الواجهات

يمكن أن نعرف واجهة لأشخاص تحتوي على بعض الخصائص كما يلي

كود

```
Interface IPerson
    Property Name() As String
    Property Birthdate() As Date
    ReadOnly Property Age() As Integer
End Interface
```

أو يمكننا تعريف واجهة لبعض العمليات الحسابية

كود

```
Interface ISomeMath
    Function AddNumbers(ByVal a As Integer, ByVal b As Integer) As
Integer
    Function AddNumbers(ByVal a As Double, ByVal b As Double) As
Double
```




```
Function Multiply(ByVal a As Integer, ByVal b As Integer) As Double

End Interface
```

كما يمكن أن نعرف واجهة لدفتر الهواتف ترث واجهة الأشخاص كما يلي

كود

```
Interface IPhonebook

    Inherits IPerson


    Property Phone() As String

    Property Address() As String

    Sub ShowInformations()

End Interface
```

الآن إن كانت لدينا فئة للهواتف نريد منها أن تحقق الواجهة Phonebook نستخدم الكلمة Implements تماما بعد التصريح عن تلك الفئة وسنرى أن بيئة التطوير ستضيف الهيكل الأساسي لعناصر تلك الواجهة إلى الفئة الجديدة

كود

```
Public Class Phones

    Implements IPhonebook

    Public ReadOnly Property Age() As Integer Implements IPerson.Age
```




```
            Get
        End Get
    End Property
.......
End Class
```

كما يمكن استخدام أكثر من واجهة ضمن الفئة الواحدة

كود

```
Class SomeTest
    Implements IPerson
    Implements ISomeMath

    Public ReadOnly Property Age() As Integer Implements IPerson.Age
        Get

        End Get
    End Property
.......
    Public Function AddNumbers(ByVal a As Double, ByVal b As
Double) As Double_
                Implements ISomeMath.AddNumbers
```



```
End Function

.......

End Class
```

ويبقى عليك كتابة الكود المناسب لتلك العناصر بما يتناسب مع وظيفة الفئة التي تعمل عليها
لاحظ تعريف جميع العناصر المضافة باستخدام محدد الوصول Public في هذه المرحلة

وهذا مثال على تعشيش الواجهات بداخل بعضها

كود

```
Interface IPhonebook
    Interface IPersone
        Property Name() As String
        Property Birthdate() As Date
        ReadOnly Property Age() As Integer
    End Interface

    Property Phone() As String
    Property Address() As String
    Sub ShowInformations()
    Event SomeEvent(ByVal a As Int16(
End Interface
```



واستخدامها ضمن الفئة

كود

```
Class test
    Implements IPhonebook
    Implements IPhonebook.IPersone

…….

End Class
```



الأصناف المجردة Abstract

صنف المجرد هوا كغيره من الأصناف يحوي توقيع فقط من الاكواد أي يحوي اسم يدل على معنى ولكن هذا الاسم لا يمكن أن يفعل شئ دون تحقيق له ...

بمعنى آخر فليكن أحدكم اسمه "همام" هل سيكون فعلا همام ومغوار دون تدريب من احد غيره ..هكذا الفئة أو الصنف المجرد ..عبارة عن أسامة سوف تعبر عن مضمون ..لكن أين ستعبر ..ستعبر في الصنف الوارث لذلك الصنف المجرد

بمعنى اشمل هوا مجرد مفهوم سوف يحول إلى حقيقة من خلا عملية implement

لحظة نحن لا نتكلم عن الواجهات interface ولكن كلمة وتقال ...دعونه على العربية كي لا احد يفهمنا غلط ..سوف نحقق الصنف المجرد في صنف ابن أخر

على فكرة ليست الأصناف الوحيدة التي يمكن أن تكون مجردة بل الــــmethod أيضا

```
abstract class PRINT
{
int X;
PRINT(int y){X=y;}
abstract void print();
}
class IMPPRINT extends PRINT
{
public void print()
{
System.out.println(super.X);
}
}
public class main
{
public static void main(String [] arg)
{
PRINT p=new PRINT()//error
IMPPRINT i=new IMPPRINT();
i.print();
}
}
```



كما تلاحظون قمنا بذكر الكلمة المفتاحية abstract قبل تعريف الصنف
أو التابع
للدلالة على انه مجرد..
نلاحظ أن داخل الصنف المجرد أو التابع لا يوجد جسم اكواد فقط التابع الباني للصنف
وهاذي يجب أن تلاحظوها لا يمكن وضع أي كود داخل الأجسام ...وكما شاهدتم
الـــ method printأيضا كيف تم تعريفها دون قوس بداية أو نهاية أي دون جسم
ثم أتينا للصنف IMPPRINT الذي قام بتحقيق ذلك الذي بلا معنى وهوا الجرد
حيث قام بعملية الطباعة..
أما في main واهم شئ يجب أن تعرفوه لا يمكن اخذ object من الصنف المجرد
بعض الملاحظات عن الموضوع
-لا يمكن وضع تابع مجردة بداخل صنف غير مجرد
-يجب على الصنف الغير مجرد الذي يرث من صنف مجرد تنفيذ جميع الطرق حتى ولو لم يستخدمها
-من الممكن التصريح على صنف مجرد يحتوي على توابع غير مجردة
-من الممكن إن يكون الصنف الابن مجرداً حتى ولو كان الصنف الأب غير مجرد
-وفي النهاية موضوع الخطأ الذي ينتج عن عملية صناعة object منabstract class

الإدخال في لغة جافا

هناك لغات برمجة عمليات الإدخال فيها أسهل من شرب كأس من الماء نعم ..

لكن الجافا تحوي بعض التعقيد في عمليات الإدخال للمعارف فيها لا أتكلم عن تلك الإدخال من نوع Joption وإدخال من خلال صناديق حوار ..بل أتكلم بشكل عام ...

حيث تقدم لغة الجافا مجاري مؤقتة buffered للإدخال التي تؤلف مصفوفة مؤلفة من البايتات أو المحارف على حسب طلب المبرمج .

نلاحظ إن الإدخال في الجافا يتكون من ثلاث أنابيب :

–system in (وهوا الأنبوب الأول الذي يعمل على قراءة بايت واحد كل مرة)



-InputStreamReader (ويعمل على تحويل كل 2 بايت إلى حرف أو رمز)

-bufferedReader (ويعمل على تجميع هذه الحروف أو الرموز في الذاكرة المؤقتة ليعمل منها سلسلة)

لتصبح المعادلة

```
BufferedReader re=new BufferedReader
(InputStreamReader(System.in;((
```

وكما نعلم بان لغة الجافا هي لغة كائنيه التوجه بحتا في ذلك لذالك أنشأنا object من عملية القراءة للوصول للمقروء

حيث أن re هوا object من الــ class bufferreader

الآن وعند استيراد المكتبة IO من java

كود

```
import java.io;*.

class BRRead}

public static void main(String args[])throws IOException}

char c;

BufferedReader br = new BufferedReader(new

InputStreamReader(System.in;((
```



```
System.out.println(”Enter characters;(“
c = (char) br.read;()
System.out.println(c;(
{
 {
```

الكود السابق لعملية إدخال بايت واحد فعندما نريد أن ندخل بايت واحد نقوم بتعريف المتحول من نمط char

أما عملية التحويل التي حدثت في السطر 8 تحويل من أسكي إلى محرف حيث قلنا أن عملية القراءة من الكيبورد تتم من خلال محارف الاسكي

كود:

```
import java.io;*.
class BRRead}
public static void main(String args[])throws IOException}
String c;
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in;((
System.out.println(”Enter characters;(“
c =  br.readLine;()
System.out.println(c;(
{
{
```



نلاحظ استخدام التابع readLine بدل من read وذلك دليل على قراءة عدة محارف مع رموز قد تتضمن فيما بينها المحرف enter

ونلاحظ أن المتحول من نمط string .

العمليات السابقة للإدخال سوف نتعامل معها أيضا في عملية إدخال بيانات لملفات عبر الـــbuffered

لكن أيضا تستخدم لعمليات الإدخال في البرامج ..

سوف اذكر طرق أخرى أو بالأحرى هي طريقة واحد للإدخال عبر class scanner

أيضا نقوم بصناعة object منه ونستخدمه في الإدخال إليكم المثال التالي

كود

```
import java.util.Scanner;

class IN

}

public void in()

}

Scanner sc=new Scanner(System.in;(

System.out.println(”enter Host or IP;(”

String Hots = sc.nextLine;()

{

 {
```



أيضا هذه طريقة إدخال مستخدمين الclass الخاص scanner

مجالات الرؤيا في الجافا

البداية سوف نقوم بشرح مهام public

ميزة الرؤية من أي فنحن نهديه أو متحول class لعنصر أو لآي public عندما نعطي ميزة في جسم public ميزة method مثلا عندما تعطي ضمن كتلة وحدة مكان من جسم البرنامج برنامج فهي لها ميزة الاستدعاء في أي مكان

```
public class anyname
{
public void namemethod()
{
}
public static void main(String [] args)
{
anyname object=anyname();
object.namemothed();
}
}
```

هنا تلاحظ استطعنا استدعاء method العامة من جسم البرنامج

مثلا لو كان لدينا متحولات عامة

```
public class anyname
{
public int X;
public int Y;
public void namemethod()
{
// هنا X,Y تستطيع استثمار
}
public static void main(String [] args)
{
anyname object=anyname();
object.namemothed();
```




```
        }
}
```

إذا كانت المتحولات عامة تستطيع استدعائها دون شروط (مبدئيا) من اي مكان من جسم البرنامج...

أما إذا كان مجال الرؤية private فهوا عكس كل ما سبق من public له خصوصية أكثر حتى على سبيل ان private class لا تستطيع إنشاء object منه!!..

نأتي ألان إلى protected

و هذا القسم من مجالات الرؤية شبيه بالـــــ publicلكنه خاص بالوراثة

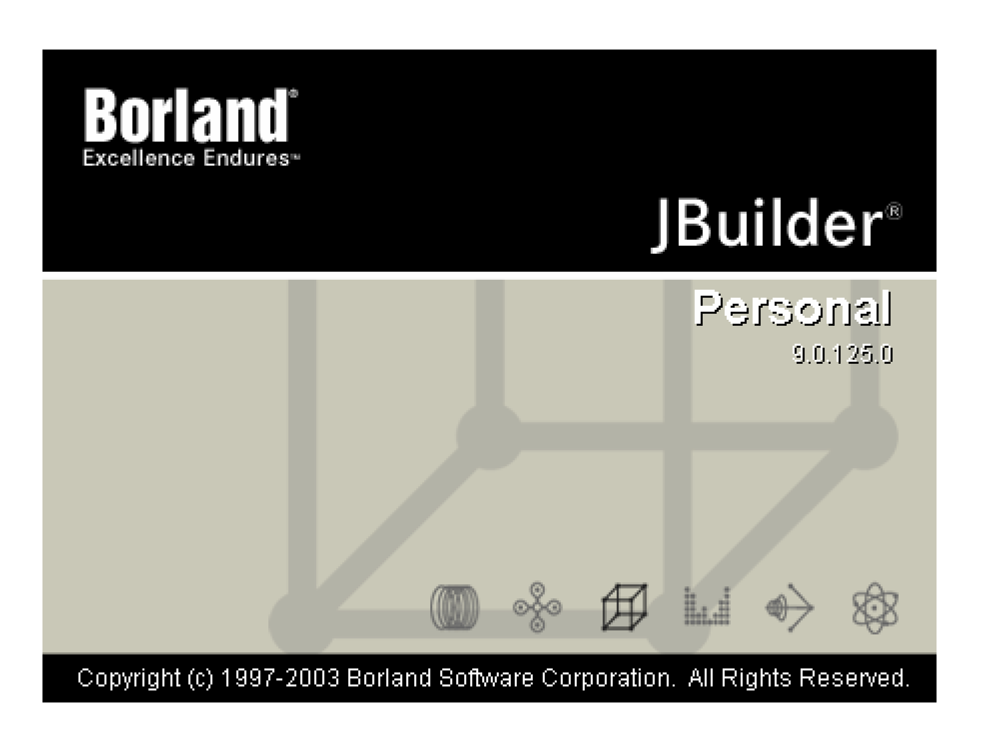
فمثلا عندما نقوم بتعريف متحول protected في class وعند اشتقاق extends من هذا الكلاس تستطيع أن تشتق المتحولات وتستثمرها أيضا إذا كانت protected

```
public class anyname
{
protected int X;
protected int Y;
}
public class anyname2 extends anyname
{
//هنا تستطيع استخدام المتحولات  X,Y
{
```



تطبيقات لغة جافا

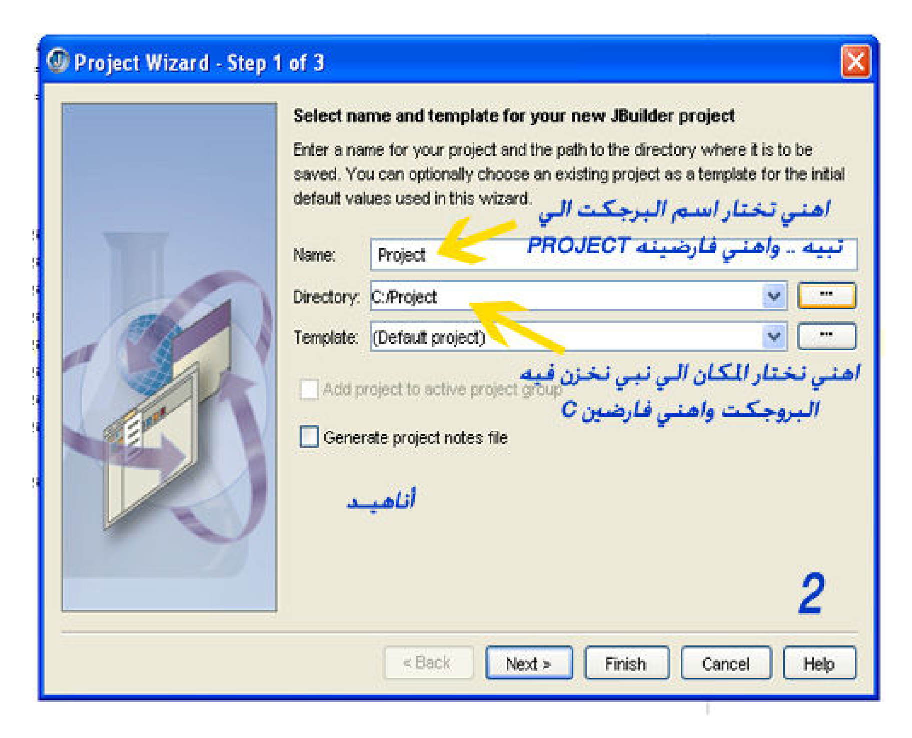
برنامج JBULDER



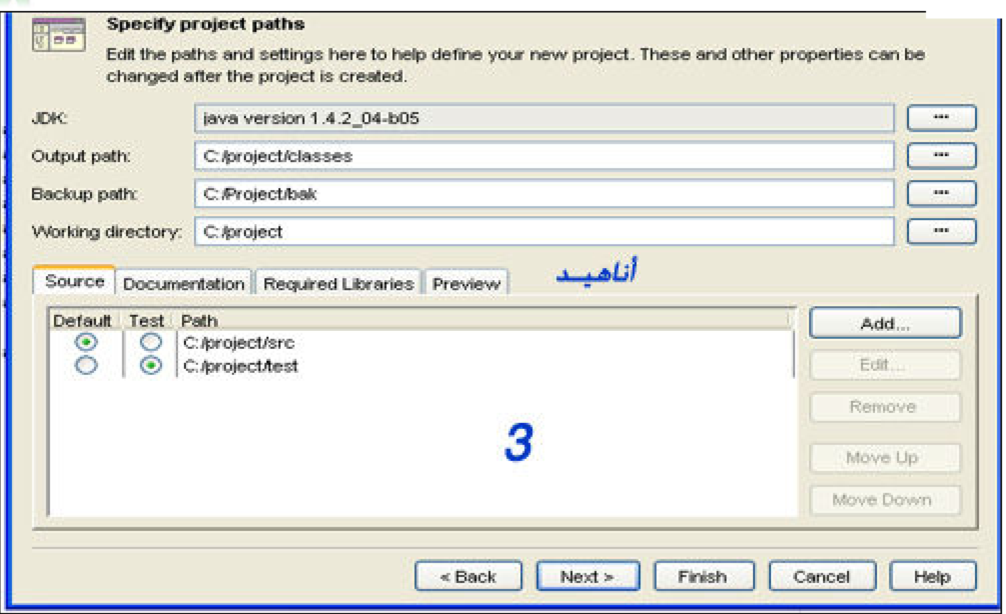
برنامج لكتابة وتطبيق برامج لغة جافا هذا البرنامج معتمد من قبل أغلب الجامعات في الوطن العربي ، حيث يمتاز بسهولة العمل عليه وإنتاج مختلف البرامج التطبيقية .

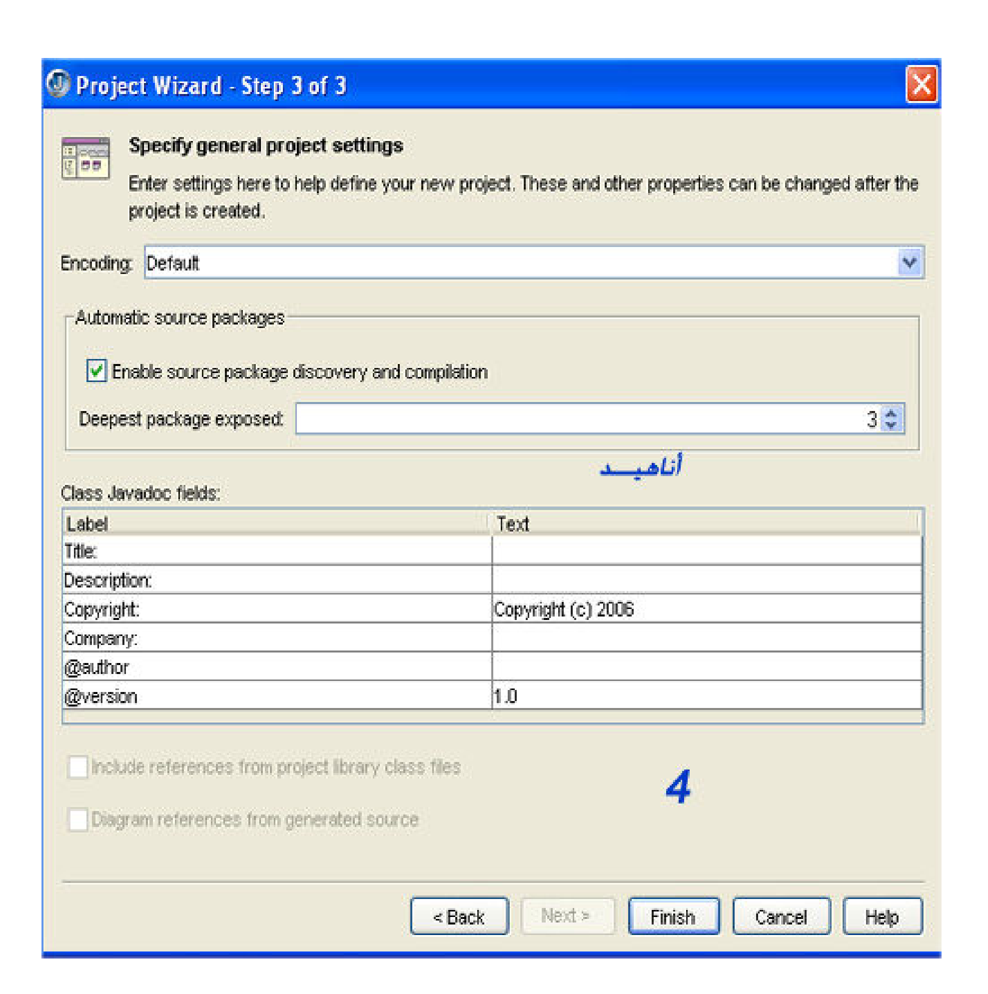



JBuilder 2005
File
Edit
Search
Refactor
View
Project
Run
Team
Enterpris
New...
Ctrl+N
New Project...
New Class...
1
New File...
أناهيد
Open Project


Project Wizard - Step 1 of 3
Select name and template for your new JBuilder project
Enter a name for your project and the path to the directory where it is to be saved. You can optionally choose an existing project as a template for the initial default values used in this wizard.
اهني تختار اسم البرجكت الي
تبيه .. واهني فارضينه PROJECT
Name:
Project
Directory:
C:/Project
Template:
(Default project)
اهني نختار المكان الي نبي نخزن فيه
البروجكت واهني فارضين C
Add project to active project group
Generate project notes file
أناهيد
2
< Back
Next >
Finish
Cancel
Help


PALTEAM


Specify project paths
Edit the paths and settings here to help define your new project. These and other properties can be changed after the project is created.
JDK:
java version 1.4.2_04-b05
Output path:
C:/project/classes
Backup path:
C:/Project/bak
Working directory:
C:/project
Source
Documentation
Required Libraries
Preview
أناهيد
Default
Test
Path
C:/project/src
C:/project/test
Add...
Edit...
Remove
Move Up
Move Down
3
< Back
Next >
Finish
Cancel
Help

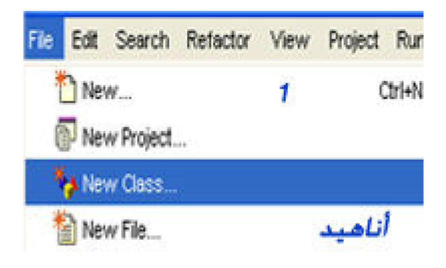

Project Wizard - Step 3 of 3
Specify general project settings
Enter settings here to help define your new project. These and other properties can be changed after the project is created.
Encoding:
Default
Automatic source packages
Enable source package discovery and compilation
Deepest package exposed:
3
أناهيد
Class Javadoc fields:
Label
Text
Title:
Description:
Copyright:
Copyright (c) 2006
Company:
@author
@version
1.0
Include references from project library class files
Diagram references from generated source
4
< Back
Next >
Finish
Cancel
Help



شرح الصور

1. الصورة الأولى :- لبدء العمل على البرنامج عليك إنشاء مشروع كما هو موضح فيها ، بالتالي عند إنشاء مشروع جديد (new project) تظهر لك الشاشة الموضحة في الصورة الثانية .
2. الصورة الثانية :- نختار منها اسم المشروع ومكان التخزين ، ويمكن أن نتركها كما هي بوضعها الافتراضي .
3. الصورة الثالثة :- نحدد من خلالها مسار المشروع ونحدد إعداداته وأين موضع تخزينه .
4. الصورة الرابعة :- يعطينا الإعدادات العامة للمشروع ومعلومات عنه ونضغط إنهاء لتظهر لنا شاشة فيها عنوان المشروع الذي أدخلناه آو الافتراضي .

بعد الخطوات السابقة نقوم بالتالي بإنشاء كلاس للبرنامج لبدء العمل والتطبيق عليه وهي كما هو موضح بالصور التالية :-







وهكذا نكون قد بدأنا العمل على البرنامج وتطبيق البرامج التي نقوم بإنشائها



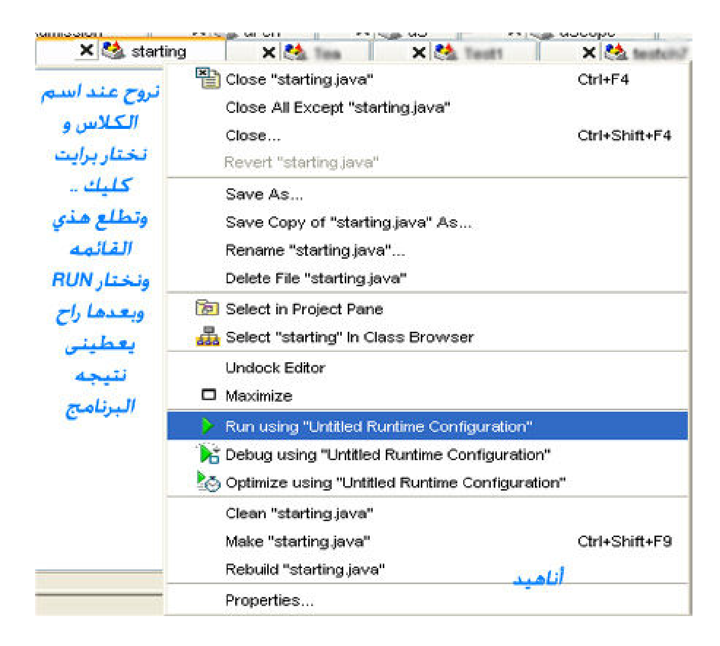
مثال على عمل برنامج وتطبيقه مثلا برنامج يطبع على الشاشة HELLOW WORDكما هو موضح بالصور التالية :-

```
 * <p>Description: </p>
 *
 * <p>Copyright: Copyright (c) 2006</p>
 *
 * <p>Company: </p>
 *
 * @author not attributable
 * @version 1.0
 */
public class starting {

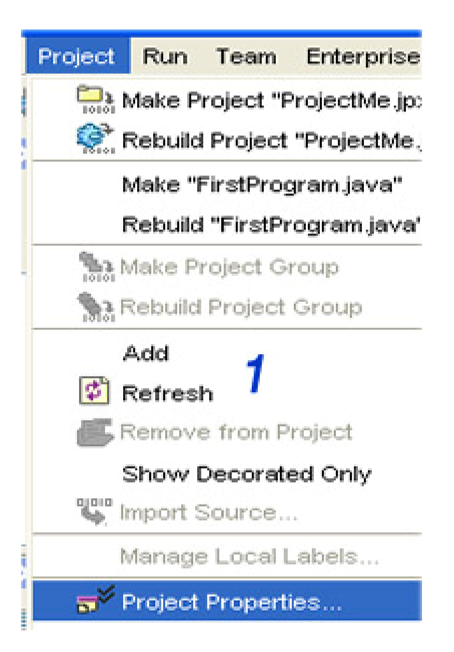
  public static void main ( String [] args){
    System.out.println( " Hello , World " );
  }

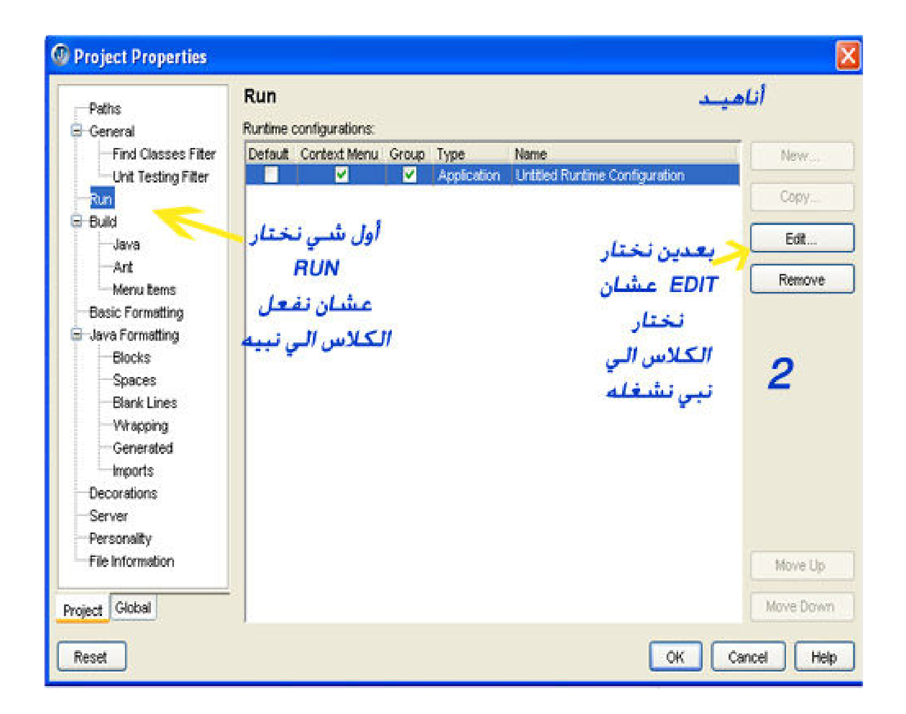
}
```

أناهيد

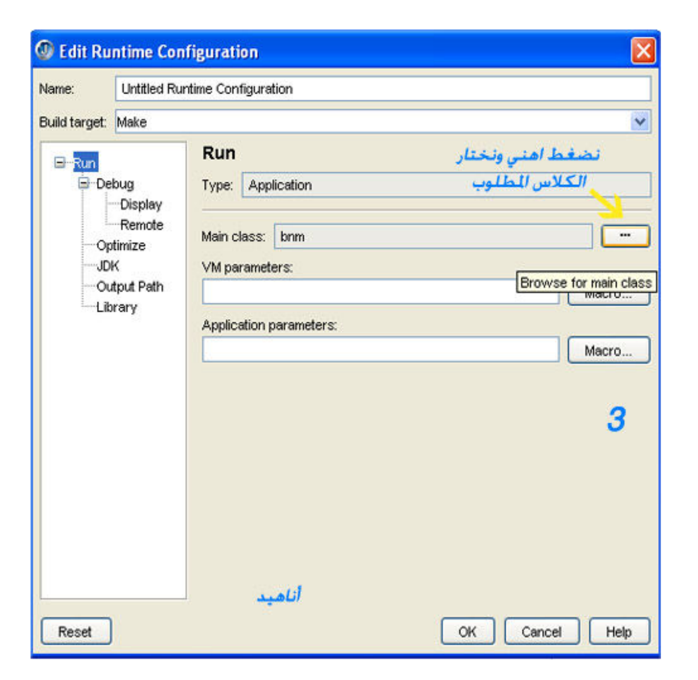





Project
Run
Team
Enterprise
Make Project "ProjectMe.jp
Rebuild Project "ProjectMe.
Make "FirstProgram.java"
Rebuild "FirstProgram.java"
Make Project Group
Rebuild Project Group
Add
1
Refresh
Remove from Project
Show Decorated Only
Import Source...
Manage Local Labels...
Project Properties...


Project Properties
أناهيـه
Run
Runtime configurations:
Default
Context Menu
Group
Type
Name
Application
Untitled Runtime Configuration
New...
Copy...
Edit...
Remove
Move Up
Move Down
Paths
General
Find Classes Filter
Unit Testing Filter
Run
Build
Java
Ant
Menu Items
Basic Formatting
Java Formatting
Blocks
Spaces
Blank Lines
Wrapping
Generated
Imports
Decorations
Server
Personality
File Information
Project
Global
Reset
OK
Cancel
Help
أول شـي نختار
RUN
عشان نفعل
الكلاس الي نبيه
بعدين نختار
EDIT عشان
نختار
الكلاس الي
نبي نشغله
2







وبعدها بيكون البرنامج على النحو التالي ويطبع :-

```
C:\Borland\JBuilder2005\
 Hello , World
```

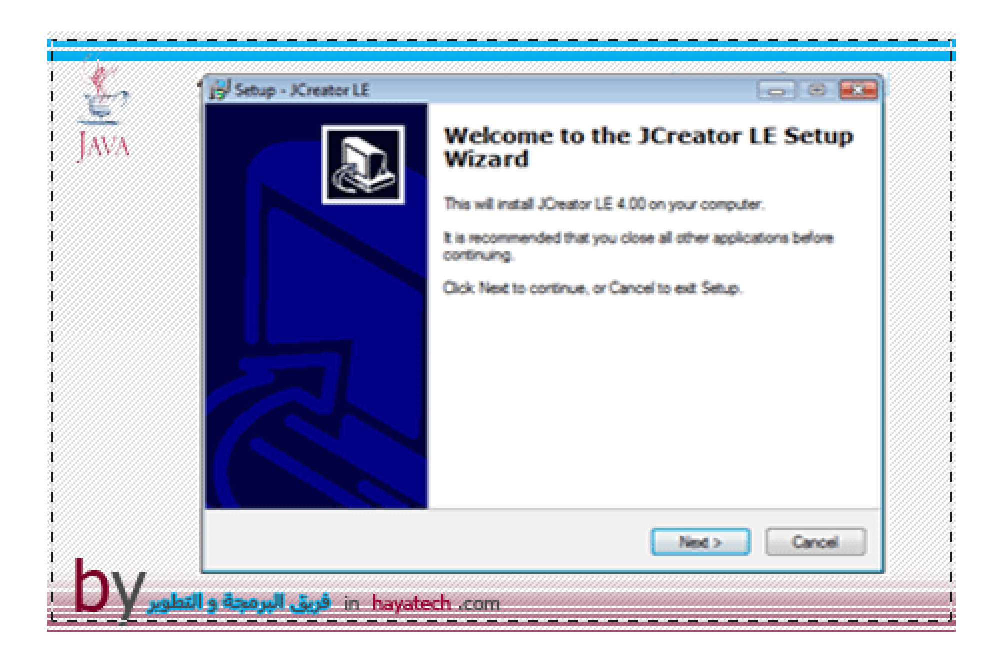
أناهيد



برنامج JCreator

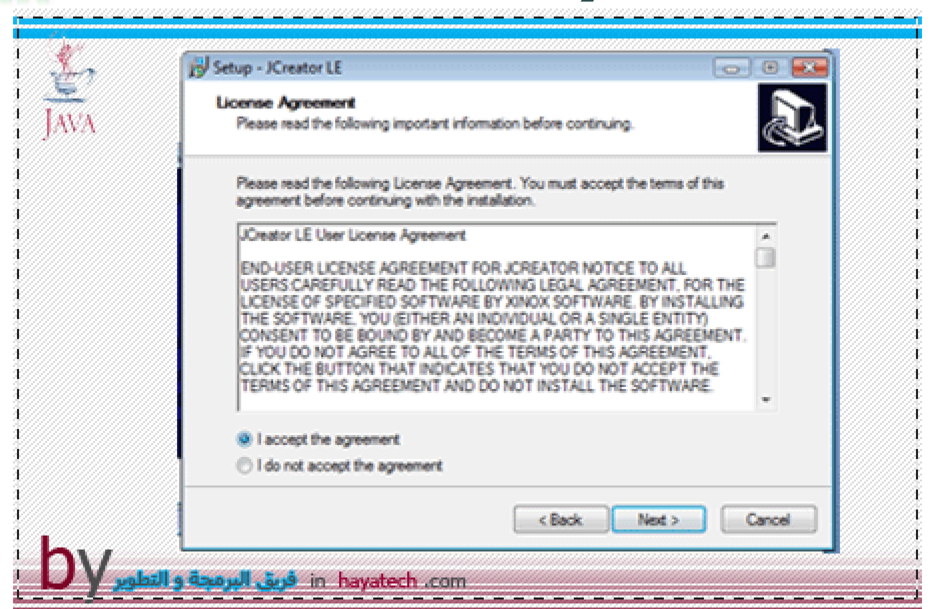


التنصيب

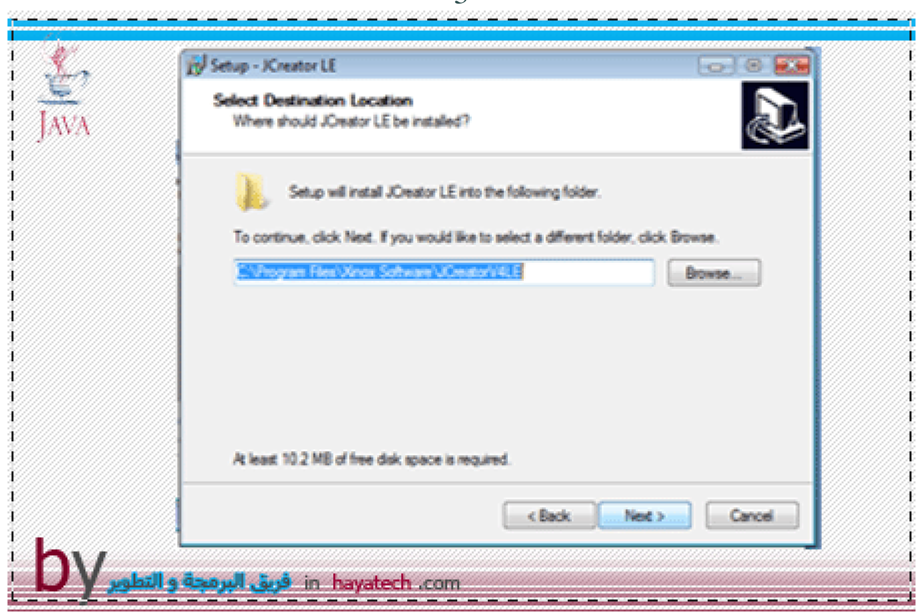




2

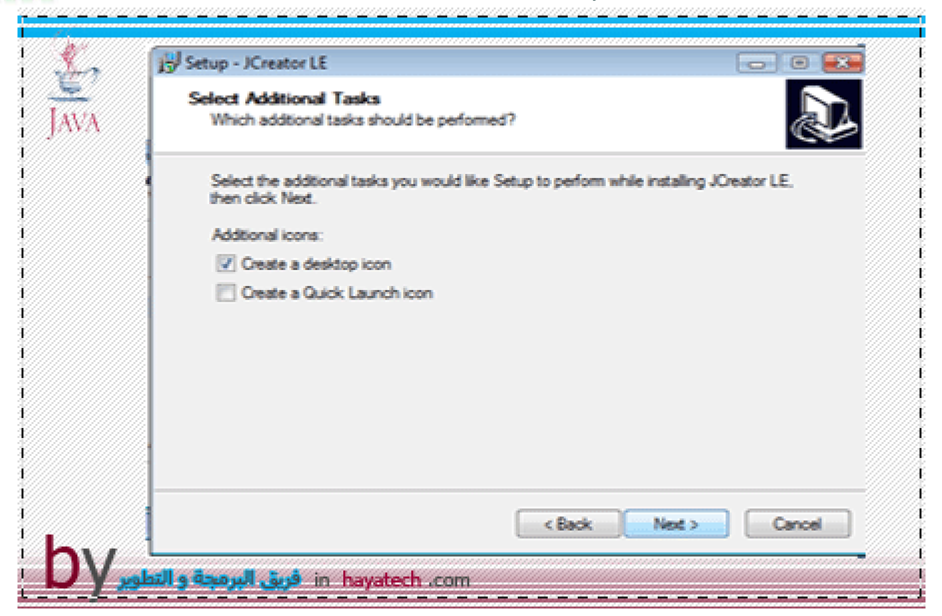

Setup - JCreator LE
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
JCreator LE User License Agreement
END-USER LICENSE AGREEMENT FOR JCREATOR NOTICE TO ALL USERS CAREFULLY READ THE FOLLOWING LEGAL AGREEMENT. FOR THE LICENSE OF SPECIFIED SOFTWARE BY XINOX SOFTWARE. BY INSTALLING THE SOFTWARE, YOU (EITHER AN INDIVIDUAL OR A SINGLE ENTITY) CONSENT TO BE BOUND BY AND BECOME A PARTY TO THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE BUTTON THAT INDICATES THAT YOU DO NOT ACCEPT THE TERMS OF THIS AGREEMENT AND DO NOT INSTALL THE SOFTWARE.
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel
by فريق البرمجة و التطوير in hayatech .com

3

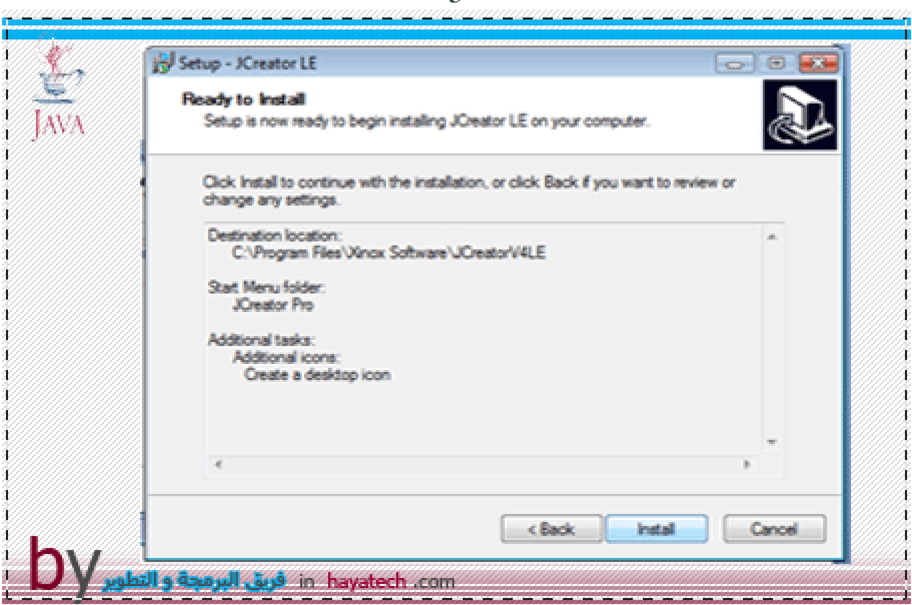

Setup - JCreator LE
Select Destination Location
Where should JCreator LE be installed?
Setup will install JCreator LE into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Xinox Software\JCreatorV4LE
Browse...
At least 10.2 MB of free disk space is required.
< Back
Next >
Cancel
by فريق البرمجة و التطوير in hayatech .com


PAL TEAM

4

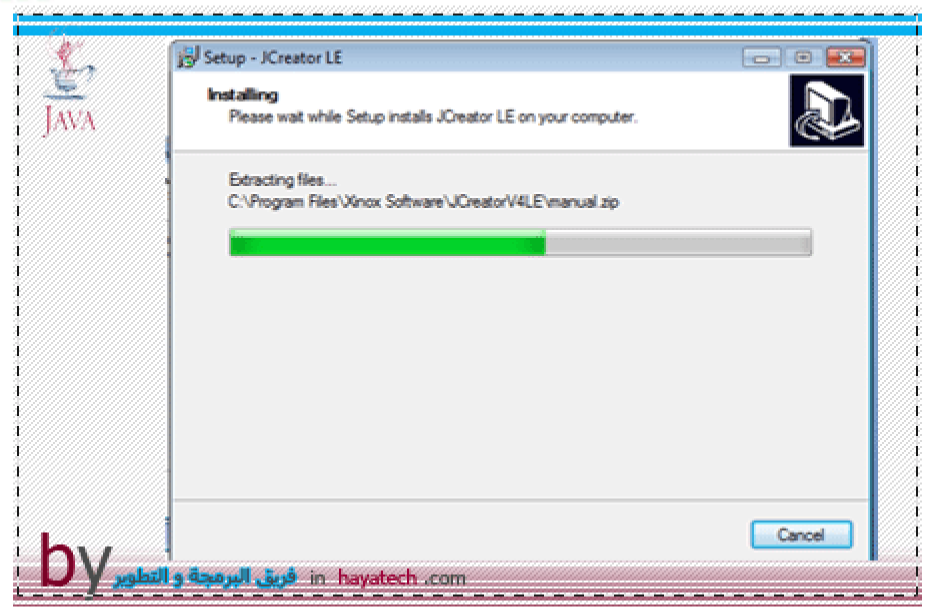

Setup - JCreator LE
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing JCreator LE, then click Next.
Additional icons:
Create a desktop icon
Create a Quick Launch icon
< Back
Next >
Cancel
by فريق البرمجة و التطوير in hayatech .com

5

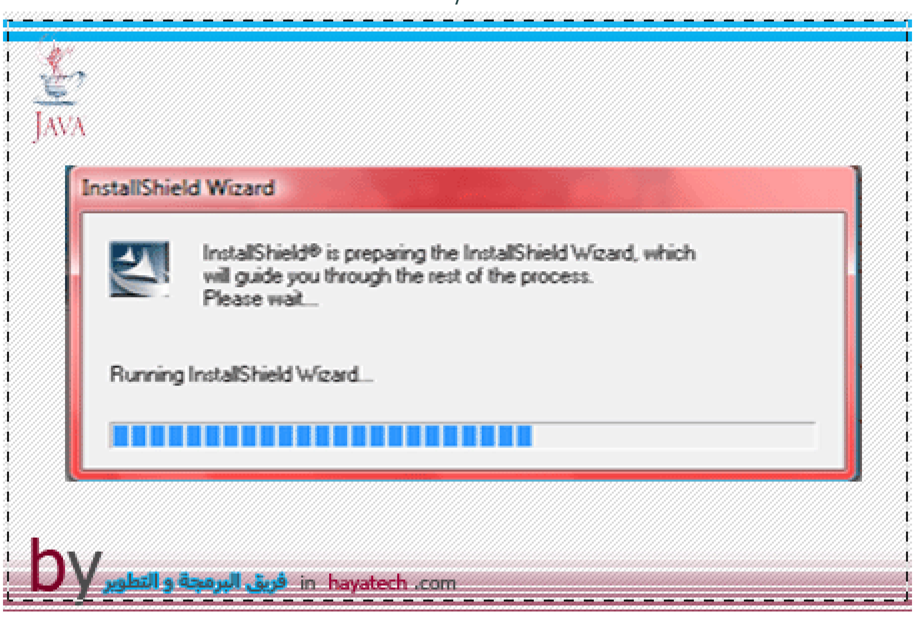

Setup - JCreator LE
Ready to Install
Setup is now ready to begin installing JCreator LE on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files\Xinox Software\JCreatorV4LE
Start Menu folder:
JCreator Pro
Additional tasks:
Additional icons:
Create a desktop icon
< Back
Install
Cancel
by فريق البرمجة و التطوير in hayatech .com


PALTEAM

6

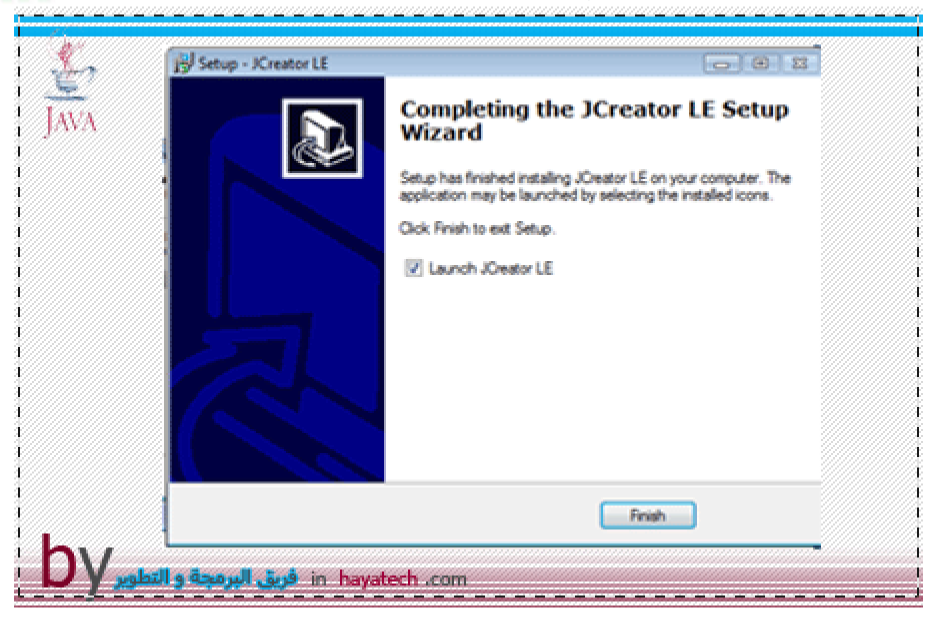

JAVA
Setup - JCreator LE
Installing
Please wait while Setup installs JCreator LE on your computer.
Extracting files...
C:\Program Files\Xinox Software\JCreatorV4LE\manual.zip
Cancel
by فريق البرمجة و التطوير in hayatech .com

7


JAVA
InstallShield Wizard
InstallShield® is preparing the InstallShield Wizard, which will guide you through the rest of the process. Please wait...
Running InstallShield Wizard...
by فريق البرمجة و التطوير in hayatech .com



8

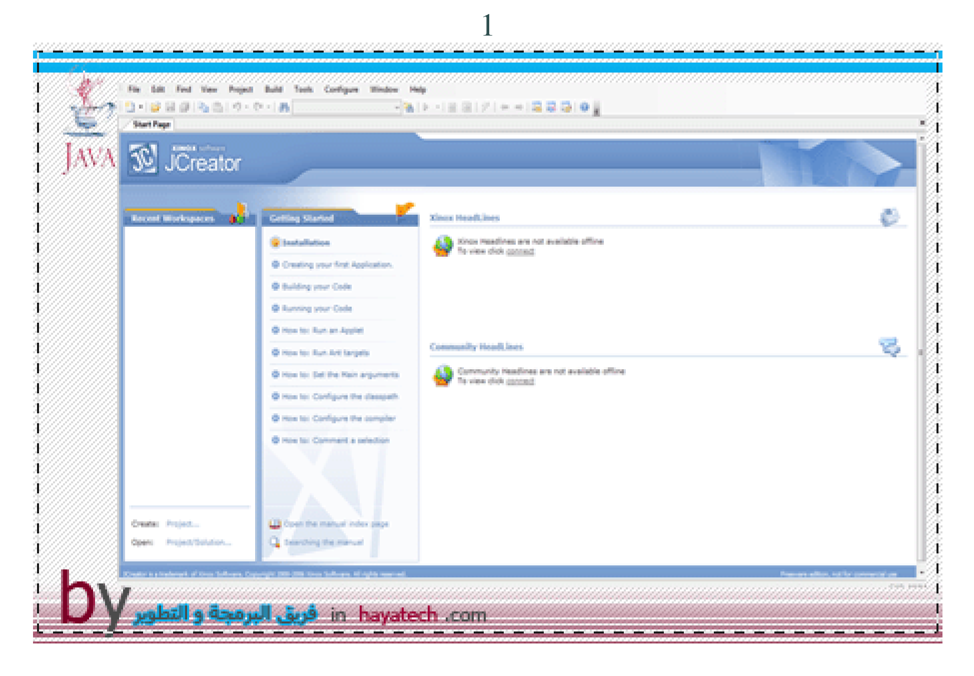


بعد التحميل نلاحظ إنشاء اختصار للبرنامج على سطح المكتب

اضغط مرتين بزر الماوس الأيسر ليفتح البرنامج

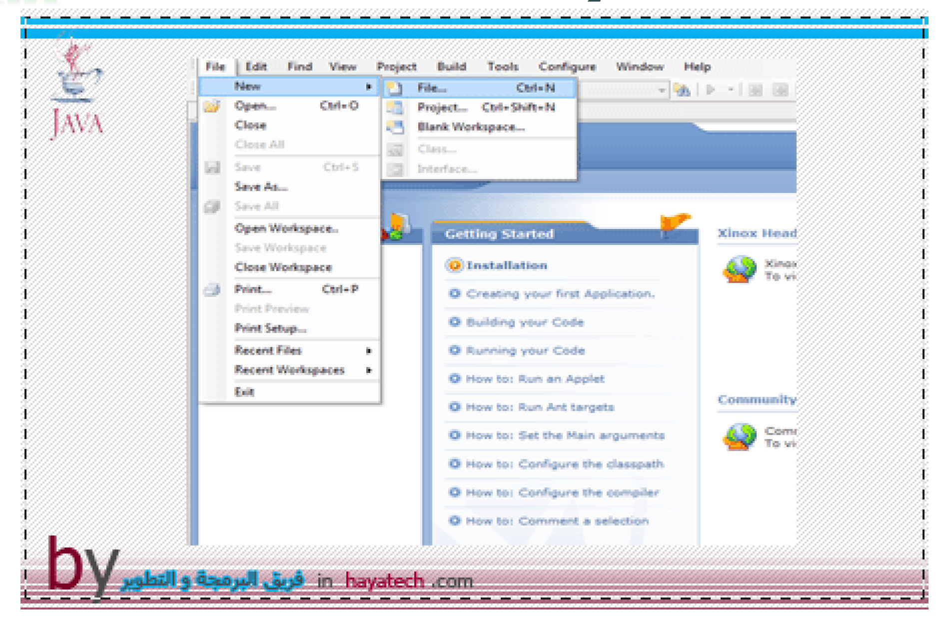
طريقة فتح المحرر و البدء بالمشروع :

1

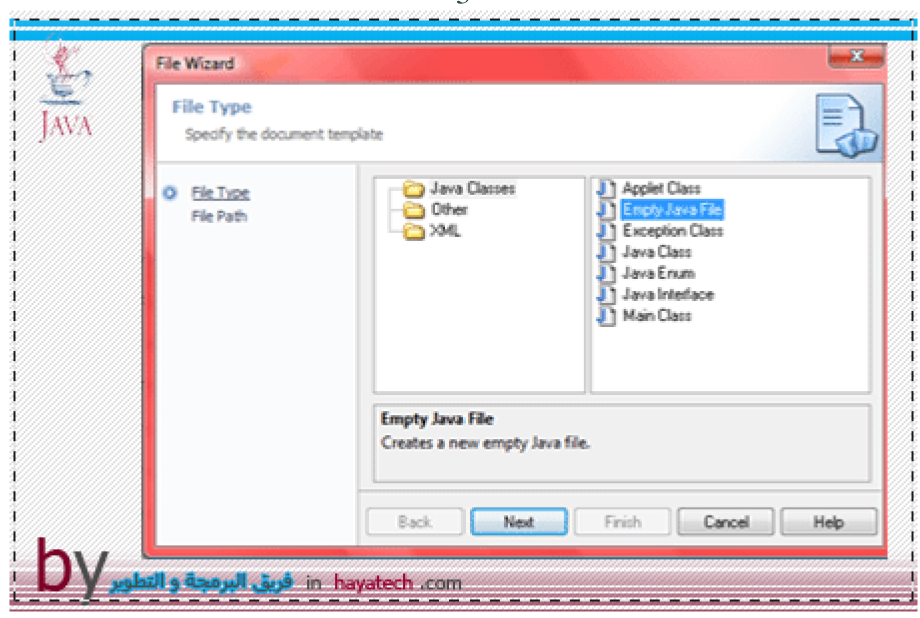




2


JAVA
File
Edit
Find
View
Project
Build
Tools
Configure
Window
Help
New
Open... Ctrl+O
Close
Close All
Save Ctrl+S
Save As...
Save All
Open Workspace...
Save Workspace
Close Workspace
Print... Ctrl+P
Print Preview
Print Setup...
Recent Files
Recent Workspaces
Exit
File... Ctrl+N
Project... Ctrl+Shift+N
Blank Workspace...
Class...
Interface...
Getting Started
Installation
Creating your first Application.
Building your Code
Running your Code
How to: Run an Applet
How to: Run Ant targets
How to: Set the Main arguments
How to: Configure the classpath
How to: Configure the compiler
How to: Comment a selection
Xinox Head
Community
by فريق البرمجة و التطوير in hayatech .com

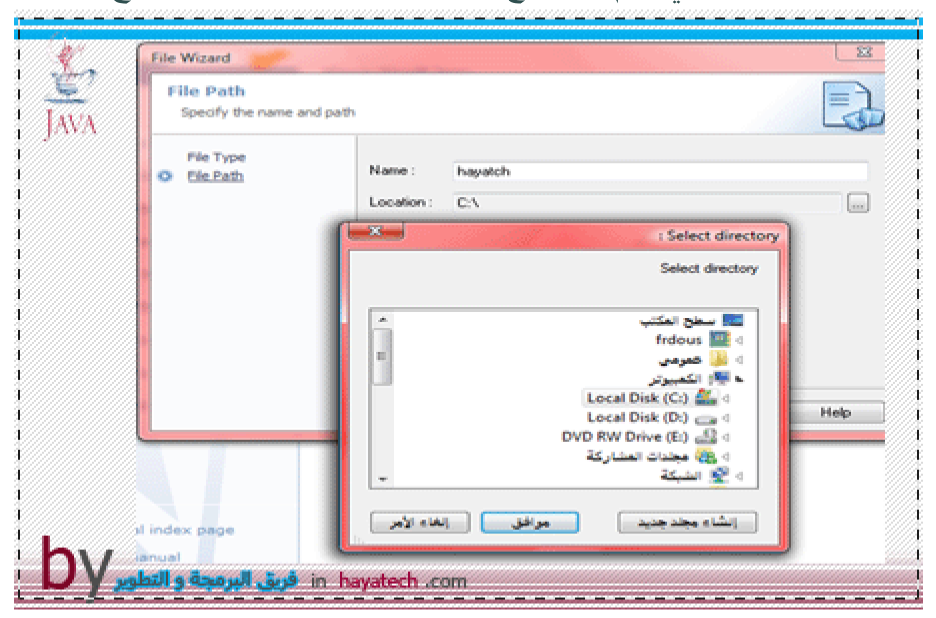
3


JAVA
File Wizard
File Type
Specify the document template
File Type
File Path
Java Classes
Other
XML
Applet Class
Empty Java File
Exception Class
Java Class
Java Enum
Java Interface
Main Class
Empty Java File
Creates a new empty Java file.
Back
Next
Finish
Cancel
Help
by فريق البرمجة و التطوير in hayatech .com

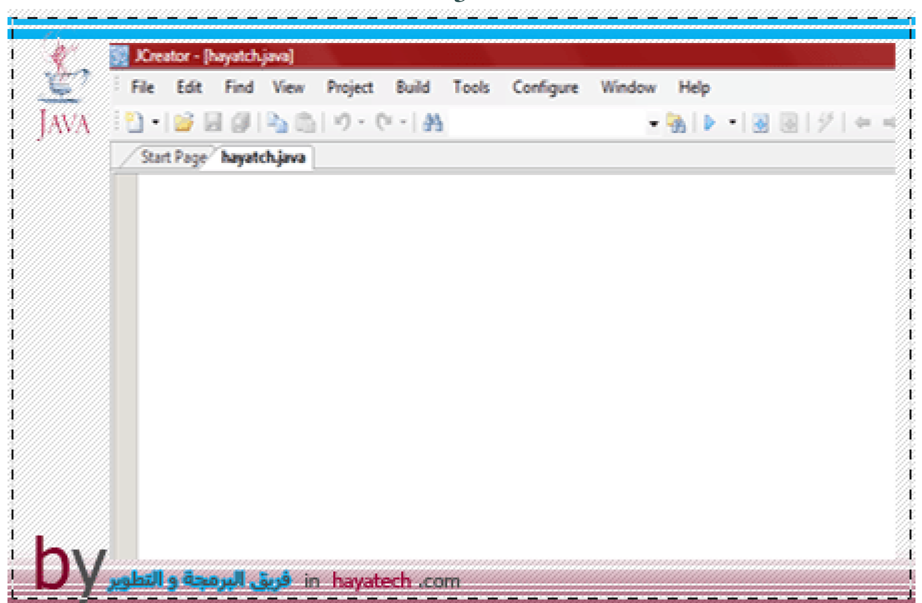


4

{ لابد أن نعطي اسم للبرنامج و مسار حفظه قبل بدء المشروع }



5



. تنفيذ البرنامج طريقة

و لمشاهدة تنفيذ البرنامج نتبع الآتي :

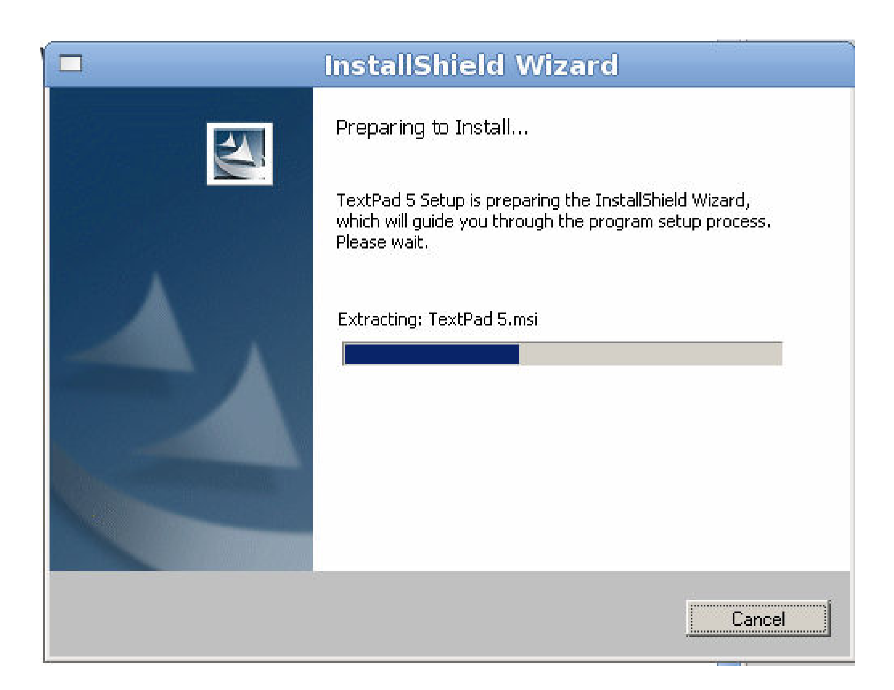
[Execute project] نختار [Build] من قائمة

أو باختصار نضغط

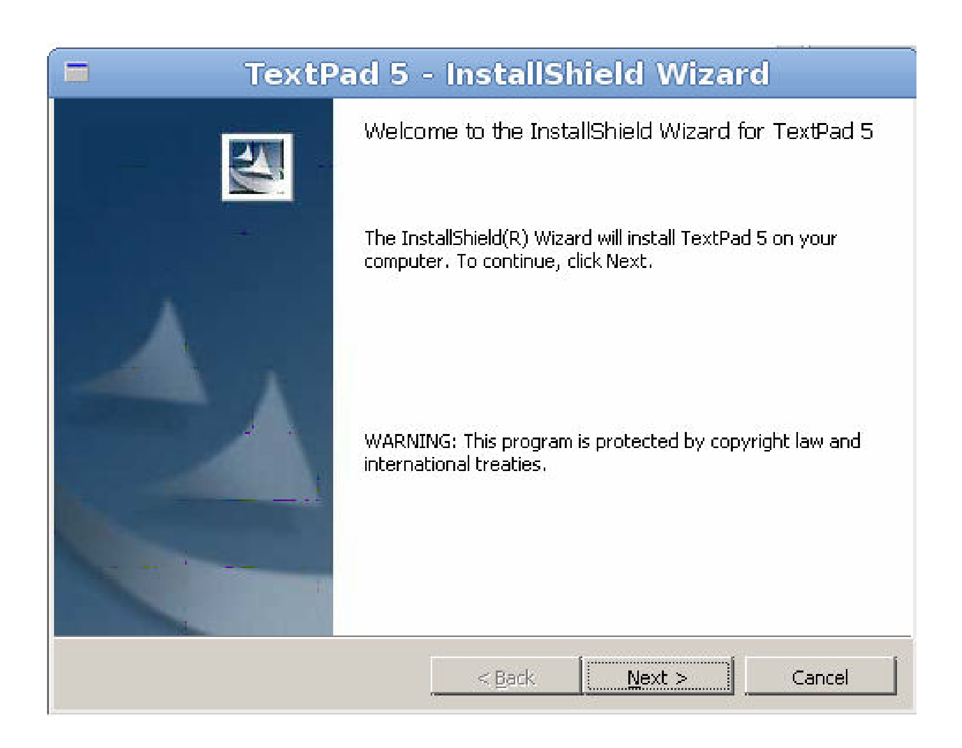
[f5]



برنامج TextPad



شاشة الترحيب الخاصة بتنصيب البرنامج اضغط Next

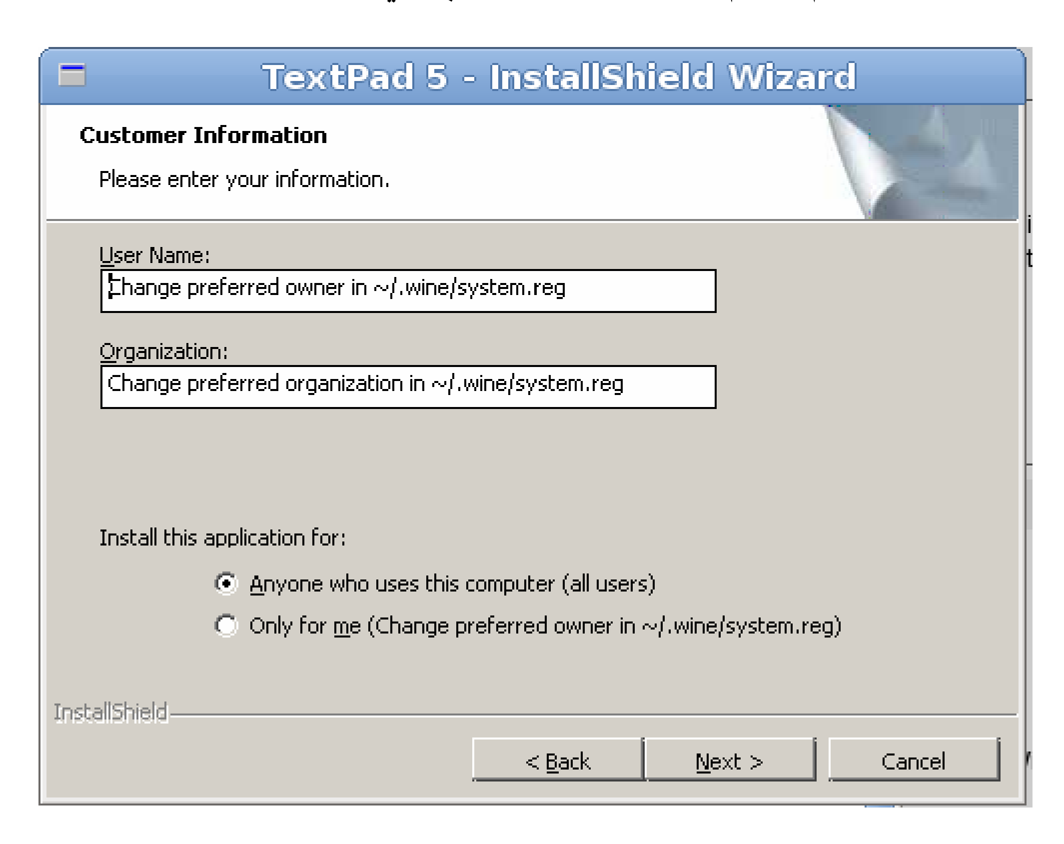




اتفاقية الترخيص يجد ة عليها ، اضغط Next

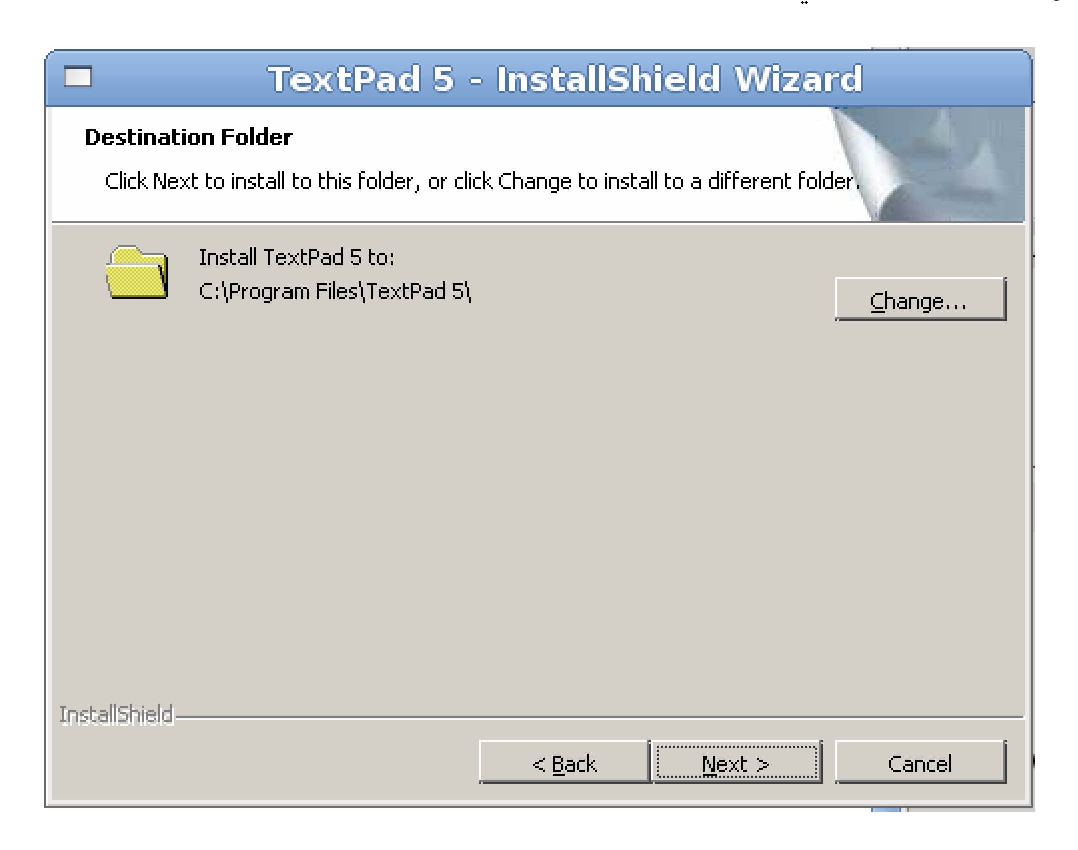


أخل بيانات المستخدم الاسم وطبيعة العمل أو أي شيء واضغط Next

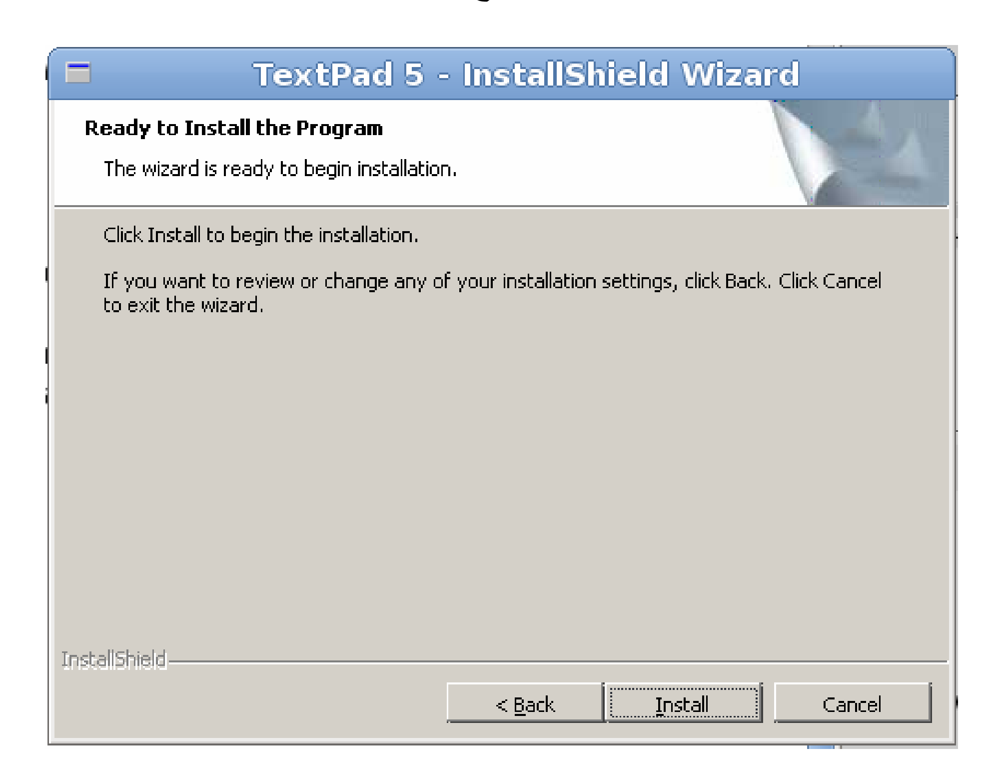




أكتب المسار الذي تريد تنصيب البرنامج عليه إن أردت من خلال
changeأو اضغط التالي ليبقى كما هو Next

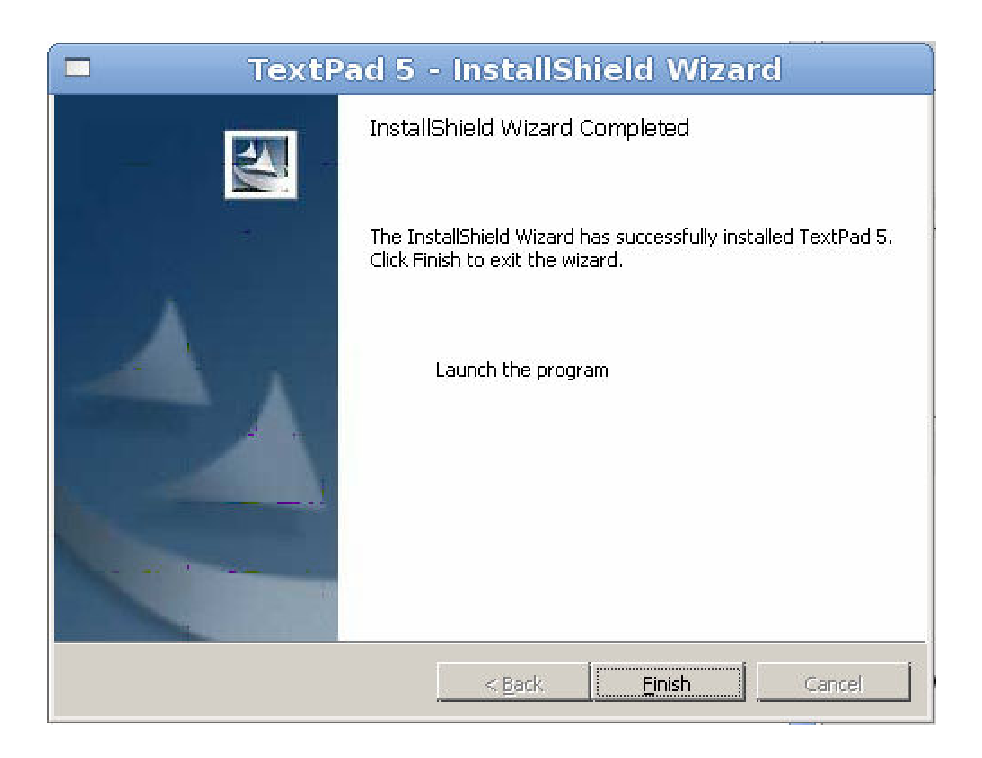


اضغط على الزر Install للبدء بتنصيب البرنامج





Finish



ستظهر إيقونة البرنامج على سطح المكتب قم بتشغيله واضغط وسيظهر لك على الشكل التالي :-

TextPad - [C:\MicroImages\SML\Kalibrierung Landsat ETM\Import_und_Kalibrie...

File Edit Search View Tools Macros Configure Window Help

```
    class FILE metFile;
    class FILEPATH metFileName$ = GetInputFileName("","Metad
    metFile = fopen(metFileName$,"r");

    string path$ = metFileName$.GetPath();
    string nameOnly$ = metFileName$.GetNameOnly();
    string tifName$ = GetToken(nameOnly$,"x",1)+"t"+GetToken

    string bandNumAll$ = PopupString("Welche Bänder sollen d
    numeric bandNumAll = NumberTokens(bandNumAll$,",");

    array numeric Lmin[10], Lmax[10];
#   array numeric gain[8];
#   class STRINGLIST gain$;
    string thisLine$, zoneNumber$, resamplingOption$;
    numeric sunElevAngle;

    # Parameter auslesen aus Met-File
    while (!feof(metFile)) {
        thisLine$ = fgetline$(metFile);

        if (thisLine$ contains "SUN_ELEVATION")     then sun
        if (thisLine$ contains "ZONE_NUMBER")       then zon
```

369 56 Read

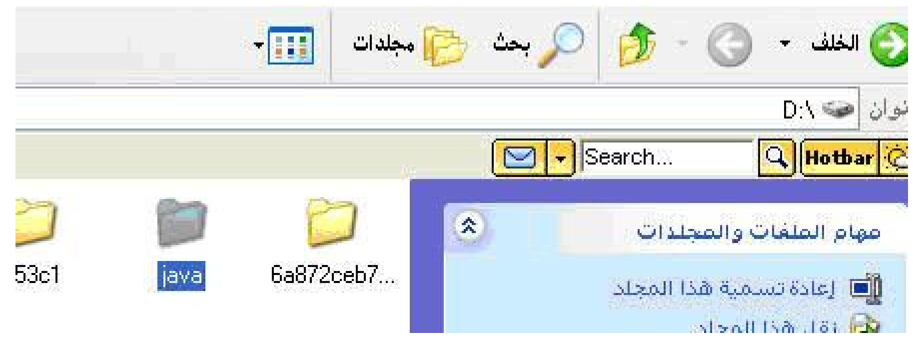


برنامج j2sedk

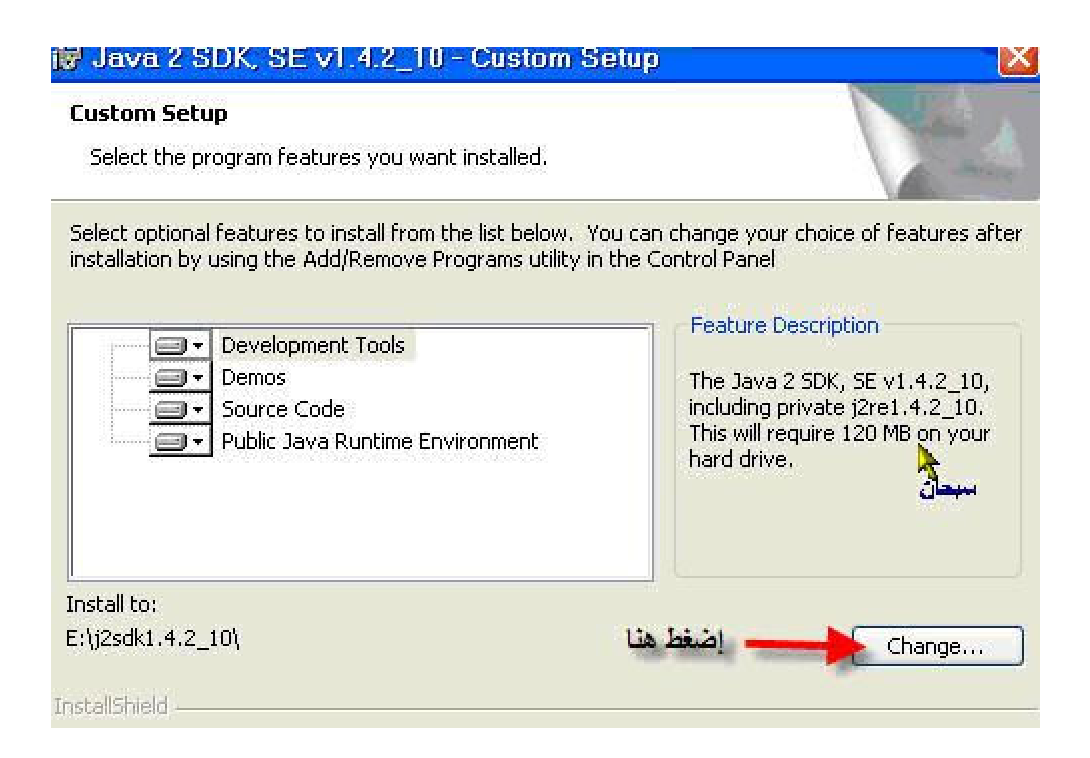
عندما تقوم بتحميل أي نسخة من نسخ j2se sdk قم بما يلي:

من أجل تنصيب مترجم الجافا قم بما يلي:

في البداية أنشئ دليلا جديدا في ال c أو في ال d بالنسبة لي أنشأت دليلا في ال d كما يلي :



بعد ذلك قم بتنصيب البرنامج المسمى j2sdk بعد بداية التحميل اضغط على changeكما في الصورة التالية:





بعد ذلك قم بتغيير اسم المجلد كما في الصورة التالية:



بعد ذلك اضغط على next كما في الصورة التالية:



بعد ذلك اضغط على install كما في الصورة التالية :



بعد ذلك للتنفيذ البرنامج أذهب إلى ابدأ ثم تشغيل ثم اكتب أمر command للوصول إلى الدوس كما في الصورة التالية:





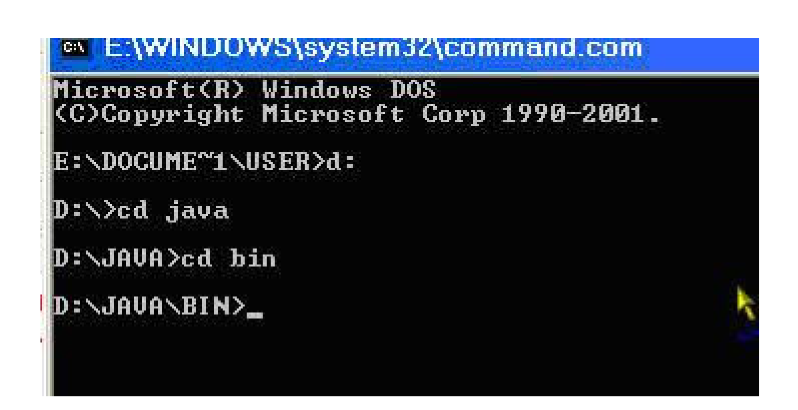
ثم بعد ذلك انتقل إلى الدليل الموجود فيه الجافا سواء كان ال c أو ال d بالنسبة لي كما قلت لكم d كما يلي:

أكتب:d: (بالعادة التخزين يكون في القرص :c)

أكتب cd java

أكتب cd bin

كما في الصورة التالية:



إذا قمت بهذه الخطوات بشكل سليم الآن تستطيع تنفيذ خطوة تخزين البرنامج وذلك عن طريق حفظ البرنامج المسمى first. java في الدليل d:\java\bin بحيث يكون البرنامج مخزن كما في الدليل التالي:

d:\java\bin\first.java



برامج باستخدام لغة Java

برنامج يخرج قائمة بأعداد الناجحين بمعدل ممتاز وجيد جدا وجيد ومقبول والراسبين أيضا :

```
import javax.swing.*;
import java.util.ArrayList;

public class while6 {

    public static void main (String args []) {

        int i = 1;
        int c90 = 0, c80 = 0, c70 = 0, c60 = 0, failed = 0;
        ArrayList<Integer> listC90 = new ArrayList<Integer>();
        ArrayList<Integer> listC80 = new ArrayList<Integer>();
        ArrayList<Integer> listC70 = new ArrayList<Integer>();
        ArrayList<Integer> listC60 = new ArrayList<Integer>();
        ArrayList<Integer> listFailed = new ArrayList<Integer>);

        while(i != 0 ) {
           String s = JOptionPane.showInputDialog("Enter number");
           int x = Integer.parseInt(s);
           if (x == 0)
              i = 0;
           if ( x >= 90 && x <= 100) {
              listC90.add(x);
              c90++;
           } else if ( x >= 80 && x < 90) {
              listC80.add(x);
              c80 ++;
           } else if ( x >= 70 && x < 80) {
              listC70.add(x);
              c70 ++;
           } else if ( x >= 60 && x < 70) {
              listC60.add(x);
              c60 ++;
           } else if ( x < 60) {
              listFailed.add(x);
              failed ++;
           }
        }
        System.out.println("A "+ c90 + " - " + listC90);
        System.out.println("B "+c80 + " - " + listC80);
        System.out.println("C "+c70 + " - " + listC60);
        System.out.println("D "+c60 + " - " + listC80);
        System.out.println("F "+failed + " - " + listFailed);
    }
}
```



إدخال رقمين وحساب مجموع والوسط الحسابي ومجموع الأرقام الواقعة بينهما

```
import javax.swing.JOptionPane;

public class sum_count_avg
{
      public static void main(String args[])
   {
      int num1=0,num2=0,sum=0,coun=0;
      float avg=0;
      String message1=JOptionPane.showInputDialog("enter your first num:");
      num1=Integer.parseInt(message1);

   String message2=JOptionPane.showInputDialog("enter your second num:");
      num2=Integer.parseInt(message2);
   for(int i=num1+1;i<num2;i++)
      {

      sum=sum+i;
      coun+=1;

      }
   avg=sum/coun;
   String msg=String.format("the sum is:%d",sum);
   String msg2=String.format("the Count is:%d",coun);
   String msg3=String.format("the Average is:%f",avg);
   JOptionPane.showMessageDialog(null,msg);
   JOptionPane.showMessageDialog(null,msg2);
   JOptionPane.showMessageDialog(null,msg3);
   System.exit(0);
   }
}
```



أ-اكتب برنامج بلغة جافا لإدخال عدد صحيح موجب وطباعة جميع الأرقام الفردية المحصورة بين ال 1 و العدد المدخل :-

```
import javax.swing.*;
class PrintOdd{
public static void main(String[] args){
     String s = JOptionPane.showInputDialog("enter a positive integer");
     int n = Integer.parseInt(s);
     for(int i=0; i<=n; i++)
          if(i%2 == 1)
               System.out.println(i);
}}
```

ب-اكتب طريقة تقوم باستلام مصفوفة من الأعداد الصحيحة و ترجع أصغر عدد.

```
public int min(int [] arr){
     int m = arr[0];
     for(int i =1; i< arr.length;i++)
          if(m < arr[i])
               m = arr[i];
     return m;
}
```

أ- عرف الصنف **Person** حسب الشروط التالية

- لكل شخص رقم هوية **(id)** و أسم **(name)**
- عرف **بناء** لتحديد قيم ابتدائية لرقم الهوية و الأسم
- عرف الطريقة **setName** و التي تتيح تغير اسم الشخص

```
class Person{
          private int id;
          private String name;
          public Person(int id1, String
name1){
                    id = id1;
                    name = name1;
          }
          public void setName(String n){
                    name = n;
          }
}
```



ب- عرف الصنف **Employee** و الذي يرث الصنف **Person** حسب الشروط التالية:

- لكل موظف رقم هوية **(id)** و أسم **(name)** و راتب **salary**
- عرف **بناء** لتحديد قيم ابتدائية لرقم الهوية و الأسم و الراتب
- عرف الطريقة **getSalary** و التي تتيح الوصول للمتغير **salary**

```
class Employee extends Person{
        private double salary;
        public Employee(int id1, String n, double s){
                super(id1,n);
                salary = s;
        }
        public double getSalary(){
                return salary;
        }
}
```

اكتب طريقة تستلم عددين صحيحين موجبين x و y و ترجع xy مستخدما مفهوم الاستدعاء الذاتي :

```
public int power(int x, int y){
if (y==0)
return 1;
if(y==1)
return x;
return x*power(x,y-1);
}
```



أكتب برنامج بلغة جافا لإدخال نص و طباعته بالمقلوب (مثال Hello تطبع(olleH

```
import javax.swing.*;
class PrintInverse{
public static void main(String[] args){
        String s = JOptionPane.showInputDialog("enter a sentence");
        for(int i=s.length()-1; i>=0; i--)
                System.out.print(s.charAt(i));
}}
```



المراجع

- كتاب برمجة (2) (لغة جافا) ، جامعة القدس المفتوحة .
- محركات البحث على الشبكة العنكبوتية .
- منتديات المبرمجين العرب .
- موقع الجافا السوري .
- موقع شركة جافا SUN JAVA .

مع تحيات الفريق الفلسطيني لأنظمة المعلومات